Nees Generative Computergraphik

Generative Computergraphik

Von Georg Nees

SIEMENS AKTIENGESELLSCHAFT

Vorwort

Frieder Nake fragt: „Kann ein Computer Kunst erzeugen?" Und er fährt fort: „Immer noch wird oft übersehen, daß diese Frage überhaupt erst gestellt werden kann, wenn eine Definition von ‚Kunst' vorliegt. Es scheint nun keine Schwierigkeit zu sein, ‚Kunst' einmal so zu definieren, daß die Frage mit ‚ja' beantwortet wird, ein anderes Mal mit ‚nein'." (Nake 68, siehe Literaturverzeichnis.) Die Ästhetiker begegnen heute der Frage „Was ist Kunst?" mit Vorsicht. „Kunst" ist ein Oberbegriff wie „Wissenschaft". Daß die Metallurgie eine Wissenschaft ist, bestreitet gewöhnlich niemand, wie steht es aber mit der Tiefenpsychologie, der theoretischen Kosmologie, oder auch der Ästhetik? Selbst solche Ausdrücke wie „Kunstwerk" oder „wissenschaftliches Gesetz" gebraucht man deshalb lieber als Rahmenbegriffe, wobei man sich ihrer Unschärfe als auch ihrer Verankerung im allgemeinen Denken bewußt ist; dann engt man jedoch die Fragestellung ein, um Antworten formulieren zu können, über die man sich vielleicht einigen kann.

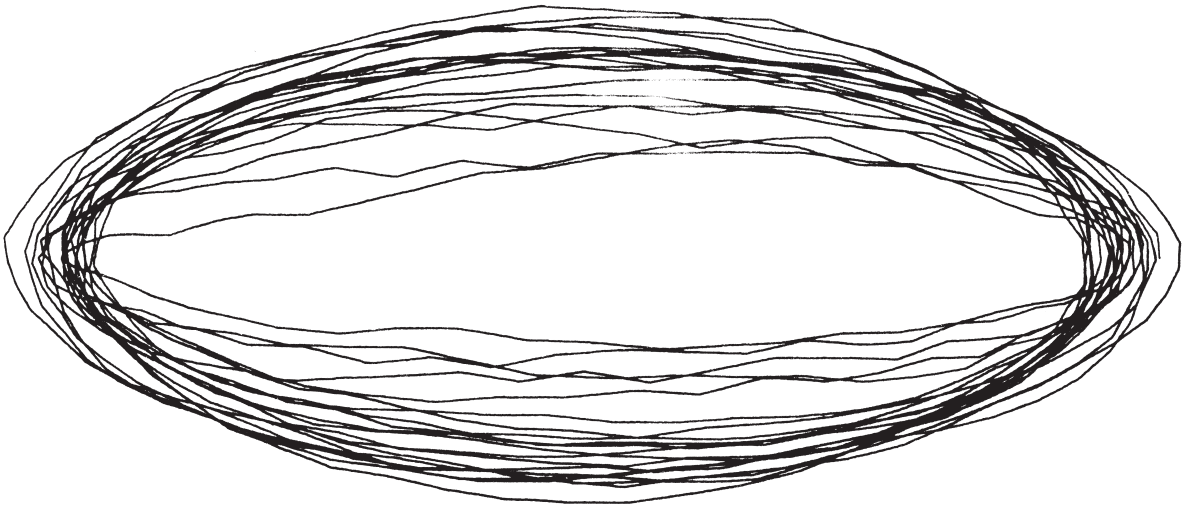
Die Ästhetik verbindet die Einengung mit einer Verschiebung der Fragestellung. In der „Semiotik" sagt Max Bense: „Auch die Kunst ist keine Aktion der Natur, sondern des Geistes. Das bedeutet, daß auch Kunst letztlich nur vom Standpunkt der Intelligenz vollständiges Interesse gewinnen kann. Sie verlangt, wie jeder Gegenstand der Welt und unseres Bewußtseins, ein gewisses Maß an Theorie, um in der Selbständigkeit und Eigenart ihrer Seinsweise verständlich zu werden. Schon frühere Ästhetiken versuchten, ein solches Verständnis zu erzielen

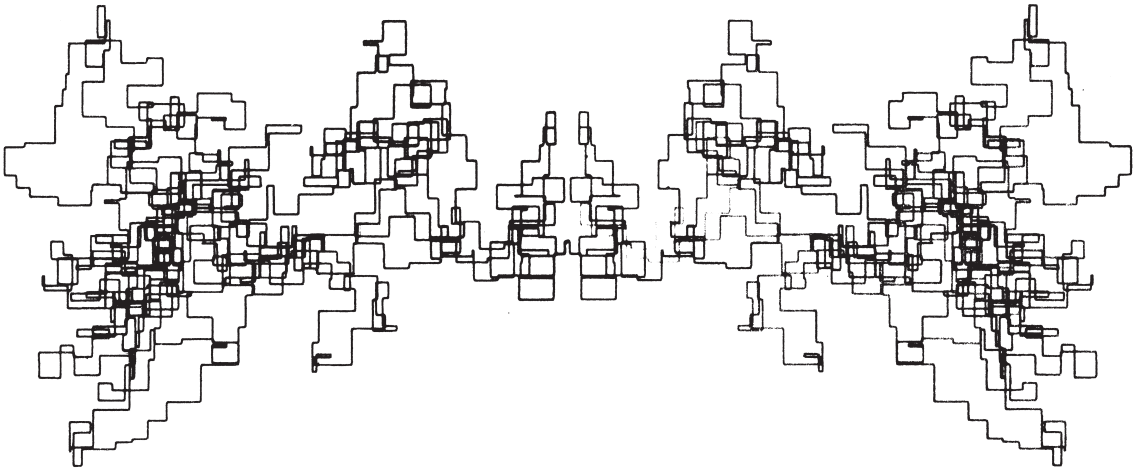
und der Kunst einen verfügbaren Ort innerhalb unseres Bewußtseins und seiner spirituellen Zusammenhänge zu verschaffen. Doch machten diese Ästhetiken ihre Begriffe und Vorstellungen von subjektiven und metaphysischen Vorentscheidungen abhängig und vernachlässigten das erforschbare und objektive materiale Sein des Kunstwerks selbst. Erst die moderne Ästhetik, die mit einigermaßen exakten Mitteln arbeitet, hat die frühere Technik der Interpretation durch eine Technik der Observation ersetzt" (Bense 67, Seite 18). Observation ist für Bense der eine Pol der Kommunikation, diese aber ist in der Ästhetik wie sonst auch der Vorgang der Übermittlung von Information von einem Zeichensender zu einem Zeichenempfänger. Ist der Zeichensender ein Bild von Manet, so ist die übermittelte Nachricht eine bestimmte ästhetische Information — und ist der Zeichenempfänger in Bereitschaft zu ästhetischer Wahrnehmung und Reflexion, so ist er Benses Observator. Was aber ist ästhetische Information jenseits vom Spezialfall? Hier wirkt sich nun die Verschiebung der Fragestellung aus: Der Ästhetiker interessiert sich zunächst nicht für die Bedeutung der Information, sondern dafür, wie sie als ein System von Zeichen aufgebaut ist. Betrachten wir ein Beispiel: In den Bildern von Manet „Der Balkon" und „Das Frühstück im Freien" erscheinen jeweils zwei Männer und zwei Frauen. Jedes der beiden Bilder ist ein komplexes Zeichen, in dem die Teilzeichen — zwei Männer, zwei Frauen — in bestimmter Anordnung auftreten. In beiden Bildern kommt ein Zeichentripel vor, im ersten Bild zwei Frauen und ein Mann, im zweiten zwei Männer und eine Frau; in beiden Bildern findet man eine Figur abgesetzt im Hintergrund. Obgleich dieses Protokoll nur der Anfang einer schrittweise verfeinerbaren und beliebig differenzierten semiotischen Analyse ist, ermöglicht es schon die Anwendung von Grundbe-

griffen der Informationsästhetik BENSES: Ein Zeichenquadrupel kehrt in zwei Bildern MANETS wieder, beide Bilder weisen Symmetrie auf — Mann Frau Frau Mann. Wiederkehr und Symmetrie wird zusammenfassend als Redundanz bezeichnet. Beide Bilder enthalten aber auch individuell Neues, das dem Observator in der Kommunikation mit dem Bild mitgeteilt wird. BENSE sagt dazu „Innovation" und bestimmt die Innovation als eine Grundkomponente der ästhetischen Information. Die Kopfhaltung der Dame im Vordergrund des Bildes „Der Balkon" ist innovativ, ja das Auftreten des Zeichens „Frau" selbst ist für dieses Bild Innovation. Innovation jedoch erscheint in einer Zeichenverteilung dispergiert und fast immer redundant. Innovation, Redundanz und Dispersion sind Grundbegriffe BENSEscher Ästhetik, die eine Theorie der Struktur ästhetischer Information ist.

In den abstrakten Bildern KANDINSKYS kehren nicht Zeichen „Mann" und „Frau", sondern Zeichen „Schachbrett" und „Kreisscheibe" wieder (siehe z.B. KANDINSKY 64). Bei MONDRIAN ist die elementare Innovation oft nur ein einziges Grundzeichen: Die Strecke. Ohne die Berufsmotivation ändern zu müssen, kann der Informationsästhetiker von der Analyse solcher artifiziellen Zeichensysteme zu der Betrachtung des Musters eines Tierfells oder von Holzmaserung übergehen; mit Hilfe eines uniformen begrifflichen Apparats wird das ästhetisch Wirksame prüfbar, wo es sich auch mitteilt. Von MONDRIAN aber ist nur ein Schritt zum Computer. Von einem Programm geleitet entnimmt der Computer Einzelzeichen aus einem Zeichenrepertoire, fügt sie aneinander, setzt sie einander entgegen, kurz: dispergiert sie und bringt sie auf ein Zeichenblatt auf, indem er die Antriebsmotoren einer automatischen Zeichenmaschine steuert. Für das Typische des Zeichenensembles, das so entsteht (wir nennen

es eine Computergraphik), ist zu einem bestimmten Teil das Programm verantwortlich, dem der Computer gehorcht. Anders ausgedrückt: Mindestens für eine Komponente der Bildinnovation zeichnet der Programmierer verantwortlich, eine zweite Komponente wird durch die Tätigkeit von Zufallsgeneratoren beigesteuert, wie in diesem Buch gezeigt werden wird. Ist die Computergraphik als komplexes Zeichen hervorgebracht worden, so kann sie in Kommunikation mit dem Betrachter — auch ihrem Programmierer — eintreten, sie kann zur ästhetischen Information werden. Das erste Bild „Computergraphiken 1965" zeigt Beispiele.

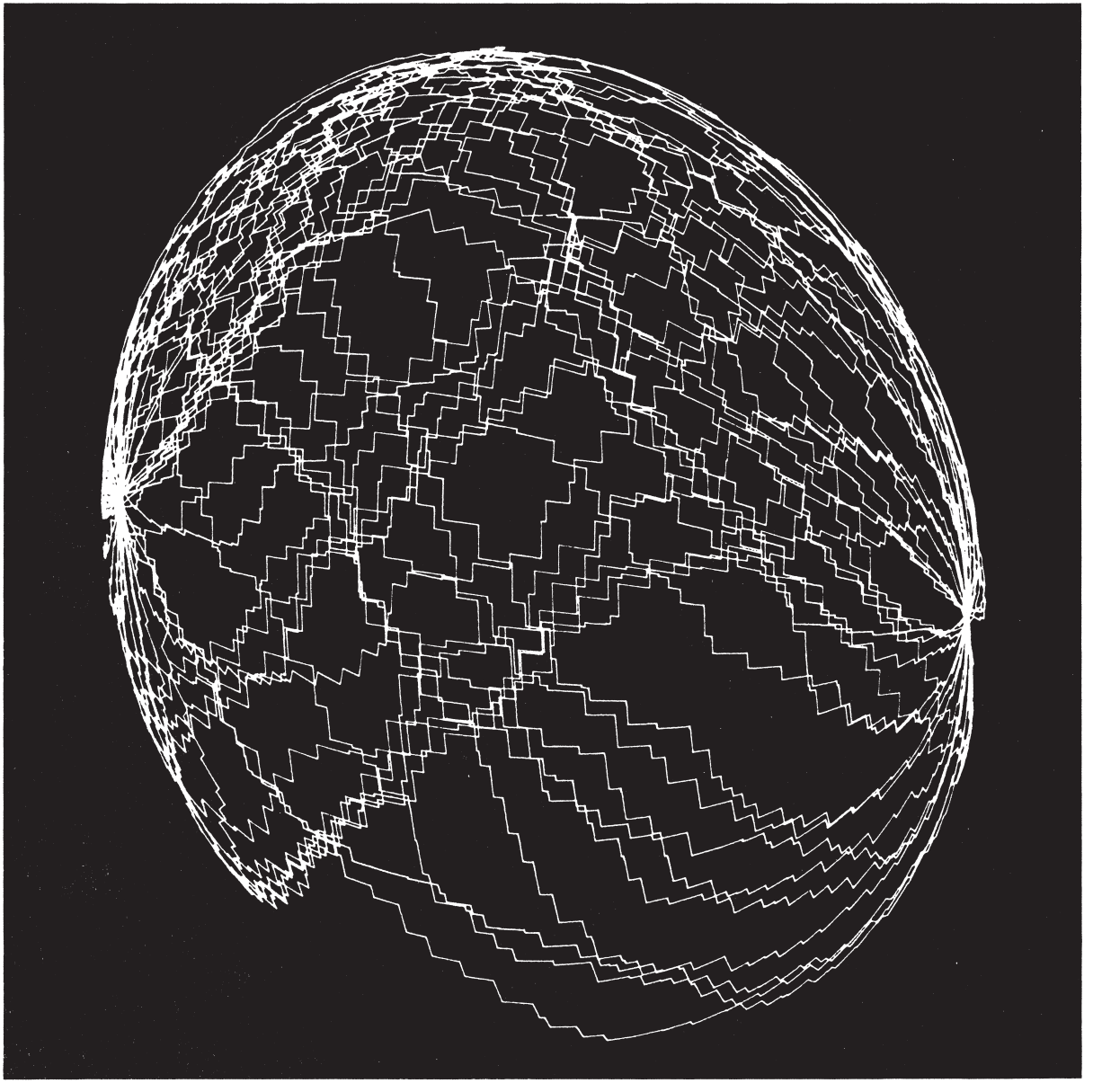




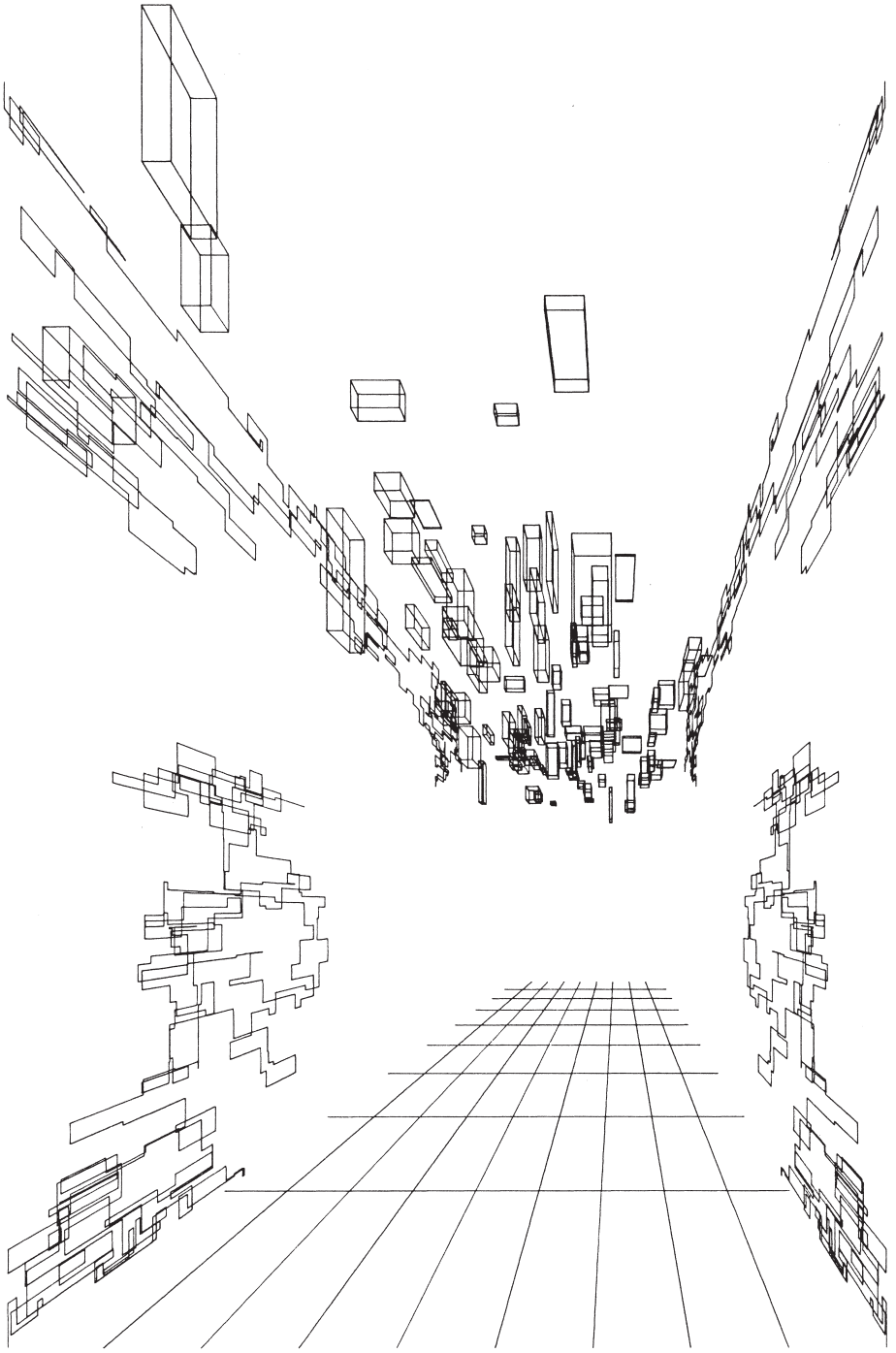
Was macht graphische Information, die der Computer erzeugt, für den Kunstwissenschaftler interessant? Um diese Frage beantworten zu können, kommen wir auf die Eigenart der Informationsästhetik zurück, daß sie sich nicht in erster Linie dafür interessiert, welche Bedeutung die Information hat, die sie untersucht, sondern für die Struktur, die diese Information als ein System von Zeichen besitzt. Nun ist der Computer sehr geeignet, beliebig umfangreiche und beliebig komplizierte Zeichen, texthafte und bildhafte, hervorzubringen, d.h. er kann dem Ästhetiker Information mit überschaubaren und in einem gewissen Umfang planbaren Eigenschaften verschaffen. Dem Kunsthistoriker gegenüber befindet sich der Informationsästhetiker dadurch in der Lage, die der Chemiker gegenüber dem Biologen einnimmt: Organische Verbindungen, die der Biologe in der Welt vorfindet, synthetisiert der Chemiker im Laboratorium — es war naheliegend, daß sich beide zusammentaten. Der Informationsästhetiker erhält also mit dem Computer und den an ihn angeschlossenen automatischen Zeichenmaschinen — und Zeichenerkennungsgeräten — ein ästhetisches Laboratorium. Die Synthesen in diesem Laboratorium können mit der Absicht der Modellierung des Stils eines Künstlers vorgenommen werden; z.B. erscheint eine Analyse per Synthetisierungen der Graphik Paul Klees mit den Mitteln der Computergraphik durchführbar, wenn man eine bekannte Graphik von Frieder Nake betrachtet (wiedergegeben in Nake 68). Ein Seitenblick auf die Computermusik macht Hoffnung: Nake berichtet in dem erwähnten Aufsatz, daß H. Kupper zweistimmige Inventionen im Stil Bachs erzeugt hat, „die von einem musikalisch wenig geschulten Hörer wohl nicht als Computermusik erkannt werden können" (siehe auch Kupper 66). Daraus läßt sich eine methodologische Forderung für die Kunstwissenschaft ableiten: Jede sprachli-

che Beschreibung einer Klasse von Kompositionen, gleichgültig, ob eine natürliche oder formale Sprache benutzt wird, muß sich in ein Programm zur Synthese von Exemplaren der gemeinten Klasse umsetzen lassen, derart, daß wir die synthetisierten Kompositionen zur Klasse zählen, wenn wir sie wahrnehmen. Dieses Prinzip macht gewisse verbale Reflexionen auf Werke der abstrakten Malerei fragwürdig, die Arnold Gehlen als „Ansingen des Bildes" bezeichnet (Gehlen 65). Jetzt kann auch eine Antwort auf die Frage Frieder Nakes gegeben werden: Selbst wenn man verneinen sollte, daß der Computer Kunst erzeugen kann, so muß man doch zugestehen, daß er als Expedient für ästhetische Information verwendbar ist.

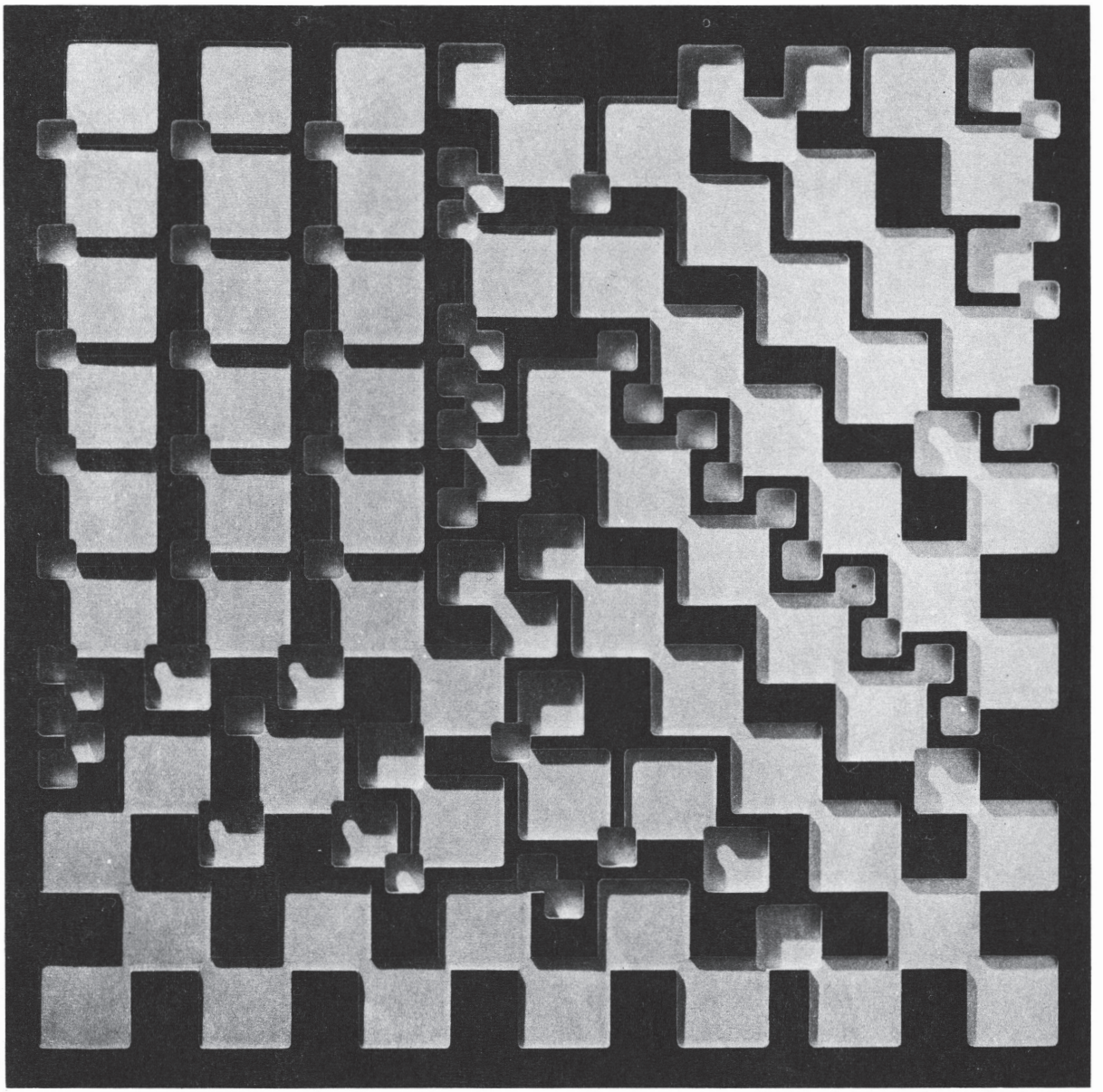
Zur Informationsästhetik, wie sie von Max Bense und seinen Schülern und von Abraham A. Moles entwickelt und von Bense insbesondere in dem Hauptwerk „Aesthetica" (Bense 65) niedergelegt worden ist, zählt auch der Inhalt des Buches, das hier vorgelegt wird. Es ist in drei Kapitel unterteilt. Nach der Einleitung im ersten Kapitel bringt das zweite Kapitel eine Anleitung zum Programmieren von Computergraphik. Diese Anleitung geht zwar anhand von Beispielen vor (man betrachte die Bilder 1 bis 14), ist jedoch so allgemein abgefaßt, daß sie jedermann, der sich praktisch mit Computergraphik beschäftigen möchte, die notwendigen handwerklichen Anfangskenntnisse vermitteln wird. An mathematischem Vorwissen werden nur die einfachsten Grundbegriffe der analytischen Geometrie vorausgesetzt. Programmieren selbst ist eine elementare Tätigkeit und dem Menschen eigentlich angeboren. Deshalb ist es richtig, den Programmierunterricht in den Lehrplan der Schulen aufzunehmen. Das dritte Kapitel ist den Strukturbeziehungen zwischen dem Programm und der von ihm ver-



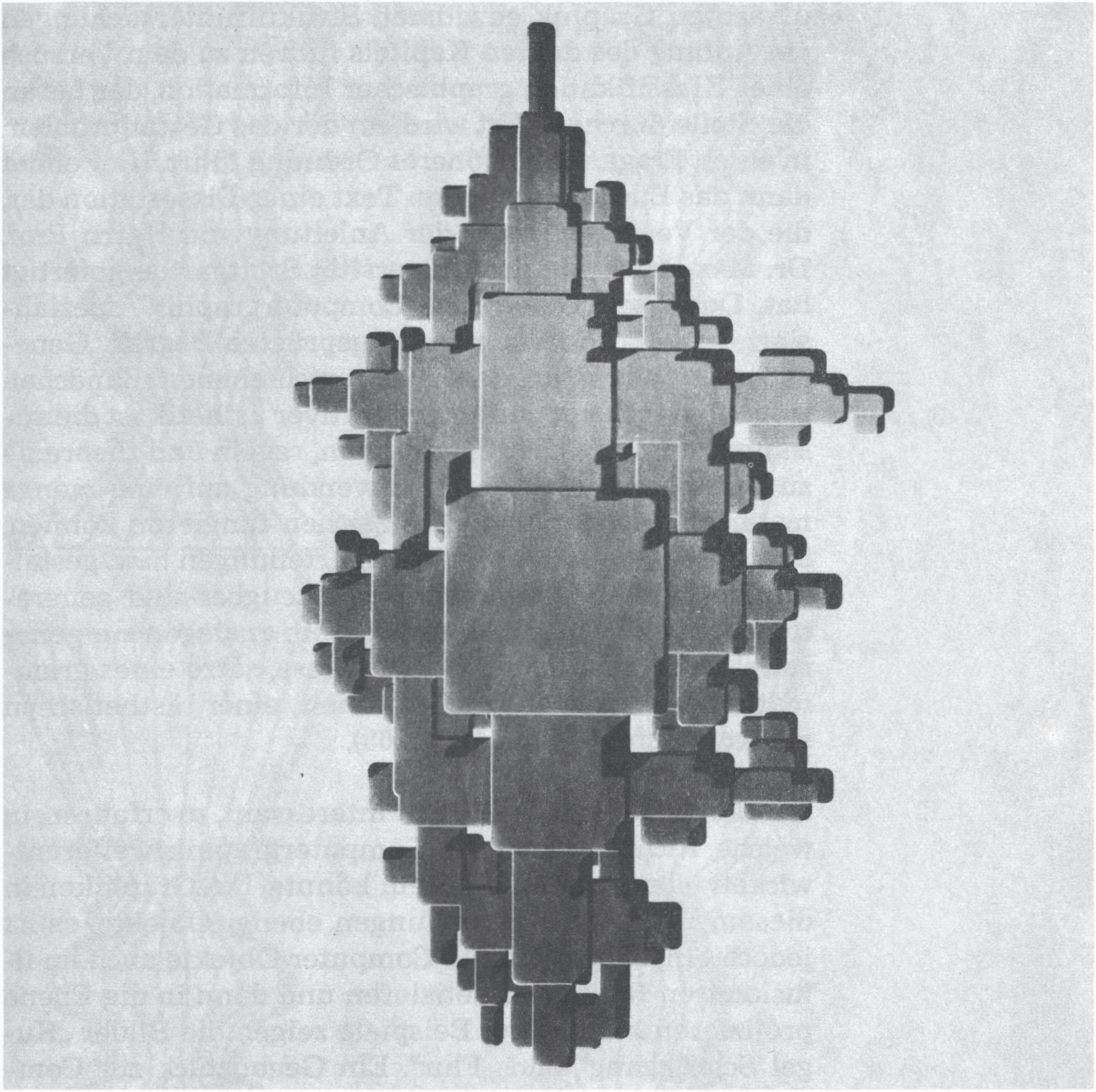
Kugel Schräggang



Flur



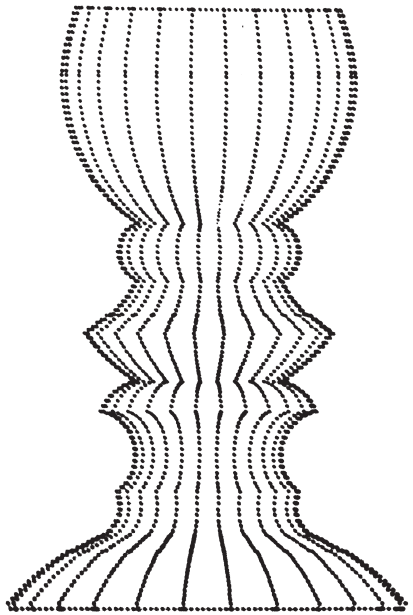
Plastik 1

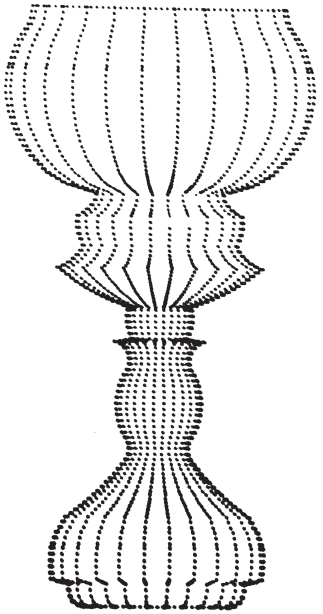


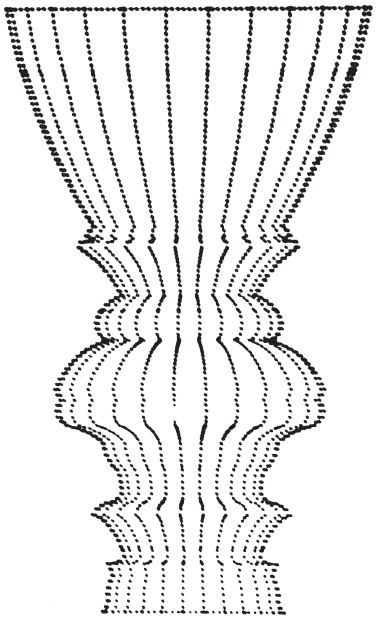
Plastik 2

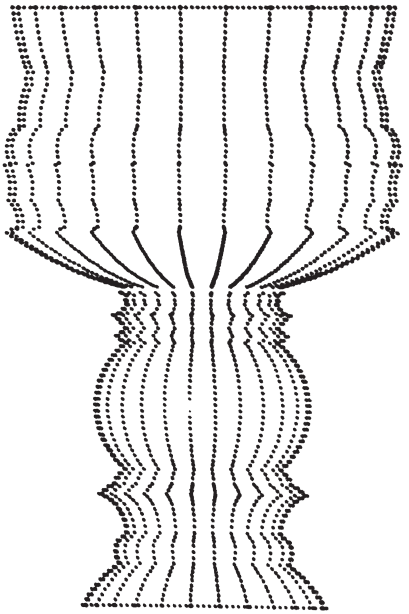
ursachten Graphik gewidmet. Strukturuntersuchungen am Anfang des dritten Kapitels führen zu dem Versuch einer Klassifikation graphischer Information, der bis an die Stelle durchgeführt wird, an der das Gestaltproblem in einen Fragenkreis höherer Ordnung führt. Hier endet dann das Buch. Es stellt den Text einer Dissertation dar, die der Verfasser unter der Anleitung von Herrn Prof. Dr. Max Bense von der Universität Stuttgart angefertigt hat. Der Titel „Generative Computergraphik" spezialisiert den von Professor Bense geprägten Begriff „Generative Ästhetik". In dem 1965 erschienenen Bändchen „rot 19" sagt Bense: „unter generativer ästhetik ist die zusammenfassung aller operationen, regeln und theoreme zu verstehen, durch deren anwendung auf eine menge materialer elemente, die als zeichen fungieren können, in dieser ästhetische zustände (verteilungen bzw. gestaltungen) bewußt und methodisch erzeugbar sind. generative ästhetik ist also in dem sinne ein analogon zur generativen grammatik, als sie, wie diese, sätze eines grammatischen schemas, realisationen einer ästhetischen struktur liefert" (Nees, Bense 65).

Für den Leser ist es vielleicht interessant, zu erfahren, in welche Richtung sich die Computergraphik weiterentwickelt oder weiterentwickeln könnte. Die Graphiken in diesem Buch sind Darstellungen ebener Objekte, es ist jedoch einsehbar, daß der Computer Objekte auch im illusionären Raum synthetisieren und dann in die Ebene projizieren kann. Zwei Beispiele zeigen die Bilder „Kugel Schräggang" und „Flur". Ein Gegenstück zur Computergraphik ist die Computerplastik. Zur Gewinnung der Plastiken in den Bildern „Plastik 1" und „Plastik 2" zog der Verfasser ein Zweistufenverfahren heran: Zuerst wurde mit Hilfe des Programmsystems, mit dem auch die Graphiken in diesem Buch hergestellt wurden,









ein Zwischenprogramm in der Form einer geeigneten Serie von Befehlen der Sprache EXAPT 1 erzeugt. EXAPT 1 ist eine Programmiersprache für die automatische Steuerung von Bohr- und Fräsmaschinen, die aus einer Zusammenarbeit von deutschen Hochschulinstituten und Firmen hervorgegangen ist (Nees 67). Aus dem Zwischenprogramm erzeugte dann das EXAPT-1-Programmsystem einen Lochstreifen, der in die Steuerung eines automatischen Fräswerks eingegeben wurde. Das Fräswerk arbeitete die Plastiken aus Holzplatten heraus.

Die Verwendung von Zufallsgeneratoren bestimmt das Verfahren zur Erzeugung der Bilder 22 bis 25 in Kapitel 3 des Buches. Frieder Nake, Michael A. Noll und der Verfasser haben diesen Weg unabhängig voneinander beschritten (siehe Nake 68 und die dort angegebene Literatur, ferner Mezei 68 sowie die Bibliographie Mezei 67). Zur ästhetischen Struktur als der Form mischen die Zufallsgeneratoren Stoff zu. Würden die Zufallsgeneratoren allein regieren, entstünde Chaos. Dadurch, daß die ästhetische Struktur durch ein Computerprogramm den Zufall einengt, in-formiert sie das Chaos. Es entsteht wahrnehmbare ästhetische Information. Der Gedanke, auf diese Weise Designserien zu studieren, liegt nahe. Ein sehr einfaches Beispiel zeigt das Bild „Vasen". Folgende ästhetische Struktur liegt hier zugrunde: Die Randkontur der runden Vasen besteht aus einer Aufeinanderfolge von Geradenstücken und Kreisbögen; eine innere Weite der Vasen wird nicht unter-, eine äußere Weite nicht überschritten. Erfahrungsgemäß zeigt jedes zehnte bis fünfzehnte Exemplar der Serie „appeal" (das Bild „Vasen" enthält solche ausgewählten Stücke). Offensichtlich sind die Intervalle, in die die Zufallsgeneratoren ihre Werte hineinstreuen, konstituierende Komponen-

ten der ästhetischen Struktur der Serie. Die ästhetische Struktur, die durch das Bild „Vasen" veranschaulicht wird, ist sehr einfach aufgebaut. Designproblemen aus der Praxis werden im allgemeinen kompliziertere, jedoch nicht prinzipiell andersartige Strukturen zugrunde liegen (s. a. Nees 69).

Zum Schluß möchte ich meinen Dank abstatten. Ich danke zuerst meinem Lehrer, Herrn Professor Dr. Max Bense, für die gute und nützliche Zusammenarbeit. Herrn Professor Dr. Walter Knödel von der Universität Stuttgart danke ich für die verständnisvolle Beurteilung meiner Arbeit. Dann gilt mein Dank dem Haus Siemens, das mir die Durchführung der Experimente ermöglichte, deren Ergebnisse in den Graphiken dieses Bandes sichtbar werden. Hier schließe ich auch den Siemens-Verlag ein, der die „Generative Computergraphik" schön ausgestattet herausgebracht hat. Frau Ute Arlt danke ich für die geduldige Herstellung der Reinschrift des Manuskripts. Nicht vergessen werden dürfen die SIEMENS-Datenverarbeitungsanlagen sowie der automatische Zeichentisch ZUSE GRAPHOMAT, ohne deren exakte Arbeit uns keine der hier gezeigten ästhetischen Informationen erreicht hätte.

G. Nees

Inhaltsverzeichnis

	Vorwort	5
1.	Einleitung	23
1.1.	Der Ort der generativen Graphik innerhalb der Informationsästhetik	23
1.2.	Übersicht über Kapitel 2	31
1.3.	Übersicht über Kapitel 3	37
	Einiges zum Abschnitt „Ästhetische Kategorien"	42
	Einiges zum Abschnitt „Gradationen"	51
	Einiges zum Abschnitt „Gestalt und Gegenstand"	59
1.4.	Weitere Probleme	60
2.	Eine Programmiersprache für die Genese von Graphiken	65
2.1.	Die Genese von Streckenkomplexen	71
2.1.1.	Die Grundanweisungen OPEN, LEER, LINE und CLOSE der Sprache G1	73
2.1.2.	Relativisierung der Lage von Komponenten einer Graphik	80
2.1.3.	Genese von Redundanz durch Laufanweisungen	85

2.1.4. Genesesteuerung durch Weichen 88
2.1.5. Genesekonvergenz und Programmschleife 97
2.1.6. Prozeduren in ALGOL und G1 102

2.2. Zufallsbehaftete Bildgeneratoren und die Sprache G2 111
2.2.1. Zufallsgeneratoren 111
2.2.2. Der Falterschwarm 118
2.2.3. Entartete Zufallsgeneratoren 123

2.3. Kreisbogen und Rechteckfläche 134
2.3.1. Die Prozedur ZIRK 135
2.3.2. Die Prozedur REC 138

3. Programm und Graphik 145

3.1. Ästhetische Kategorien 146
3.1.1. Selektoren und Kompositoren 156
3.1.2. Transduktorstruktur und Programmstruktur 161
3.1.3. Komponierende Selektion 163
3.1.4. Makroinnovation und Mikroinnovation 167
3.1.5. Redundanz und Makroinnovation 178
3.1.6. Induzierte und inhärente Innovation 180
3.1.7. Signal, Rauschen und Mikroinnovation 181

3.2. Gradationen 183
3.2.1. Ein Rahmenprogramm für eine Experimentalserie 185
3.2.2. Der elementare Irrweg 189
3.2.3. Variationen 194
3.2.4. Die Variation zwischen Haufen und Gestalt 201
3.2.5. Texturen 211

3.2.6. Lokale Innovationsdichte
Endogene Redundanz 229
3.2.7. Gewirre und Gerölle
Texturaddition 236
3.2.8. Anisotropie in Texturen 243
3.2.9. Gewebe 256
3.2.10. Ornamente 266
3.2.11. Ein Gradationsdiagramm 271

3.3. Gestalt und Gegenstand 275
3.3.1. Die gegenstandsschaffende Reflexion 281
3.3.2. Kontur und Gestalt 287

Stichwortverzeichnis 290

Namensverzeichnis 295

Literaturverzeichnis 296

1. Einleitung

1.1. Der Ort der generativen Graphik innerhalb der Informationsästhetik

Eine Einführung in den Gegenstand der generativen Graphik kann nicht besser begonnen werden als mit der Definition der generativen Ästhetik, wie sie MAX BENSE im rot-Bändchen 19[1]) folgendermaßen formuliert: „Unter generativer Ästhetik ist die Zusammenfassung aller Operationen, Regeln und Theoreme zu verstehen, durch deren Anwendung auf eine Menge materialer Elemente, die als Zeichen fungieren können, in dieser ästhetische Zustände (Verteilungen bzw. Gestaltungen) bewußt und methodisch erzeugbar sind." Betrachtet man die generative Ästhetik als ein Teilgebiet der Informationsästhetik, die sich in jeder notwendigen Weise mit dem ästhetischen Aspekt von Information, d. h. von Zeichensystemen beschäftigt, so ist das spezielle Ziel der generativen Ästhetik das Studium der operativen Herstellung von Zeichensystemen (über einem Arsenal von Zeichen), die als ästhetische perzipierbar sind. Anders ausgedrückt: Generative Ästhetik widmet sich der regularisierten Genese und der wissenschaftlichen Reflexion auf solche Genese von ästhetischer Information. Dadurch, daß ihr Gegenstand besondere ästhetische Information, nämlich graphische Information ist, sondert sich nun die generative Graphik aus der allgemeinen generativen Ästhetik aus. Unter Graphik verstehen wir ganz im klassischen Sinn die flächige Darstellung von Zeichensystemen vornehmlich in Schwarzweiß, wobei die Zeichen mit einer gewissen Diskretisation auf die Fläche gebracht werden. Die von der Graphik dargebotenen Zeichenverteilungen

[1]) Computer-Grafik, rot-19, herausgegeben von M. BENSE und E. WALTHER; Stuttgart 1965

betrachten wir als graphische Information. Fälle von graphischer Information sind die Bilder, die zu unserer Untersuchung gehören.

Schon in rot-19, jedoch etwas später auch in den Aesthetica[1]) (Seite 334) schildert Bense die Vorgehensweise der generativen Ästhetik: „Das effektive Ziel des Systems generativer Ästhetik besteht darin, die Charakteristiken ästhetischer Strukturen, die in einer Menge materialer Elemente realisierbar sind, numerisch und operationell so zu beschreiben, daß sie als abstrakte Schemata eines *Prinzips Gestaltung,* eines *Prinzips Verteilung* und eines *Prinzips Zusammenhang* gelten können und manipulierbar einer materialen, ungegliederten *(verdampften)* Menge von Elementen aufgedrückt werden können, um gemäß diesen *Prinzipien* das hervorzurufen, was wir als *Ordnungen* und *Komplexität* makroästhetisch und als *Redundanzen* und *Information* mikroästhetisch am Kunstwerk wahrnehmen. Das Aufdrücken ist indessen nicht als Anwendung einer Schablone zu verstehen, sondern als Erzeugungsprinzip. Auch *Programme* in bestimmten *Programmiersprachen* zur *maschinellen* Realisation *freier* (stochastischer, intuitiver) oder *gebundener* (im voraus festgelegter, deduzierter) ästhetischer Strukturen gehören zum System generativer Ästhetik und ihren Projekten, sofern sie metrische (Abstände, Wortlängen), statistische (Wortfolgen, Positionierungen) und topologische (Verbindungen, Deformationen) Bestimmungen einkalkulieren, um *ästhetische Information* zu erzeugen." Wir halten im Zitieren dieses grundlegenden Textes ein, um schon hier Konsequenzen für die Ausführung des *Projekts generativer Graphik* aufzuweisen. Sofern wir versuchen, graphische Information im

[1]) Max Bense: Aesthetica, Agis-Verlag, Baden-Baden 1965; künftig zitiert als AE

methodologischen Rahmen der generativen Ästhetik sowohl zu produzieren als auch zu begreifen, muß graphische Information prinzipiell verstehbar sein, als bestimmt durch Gestaltung, Verteilung und Zusammenhang. Für jede einzelne Graphik sind daher diese drei Prinzipien zu verknüpfen mit drei Merkmalen *Gestaltung, Verteilung, Zusammenhang,* die jedoch in keiner Weise begriffsrealistisch als die der Graphik vorgeordneten Ideen aufgefaßt werden dürfen, vielmehr handfest die materielle Herstellung der betreffenden Graphik zu überwachen haben. Das kann nur bedeuten, daß schon die Vorschrift, nach der die Graphik produziert werden soll, auf Gestaltung, Verteilung und Zusammenhang hinarbeiten muß. Diese Produktionsvorschrift für die Graphik ist nichts anderes als das von Bense erwähnte Erzeugungsprinzip der Graphik: Ihr Programm. Nun kann man sich die Entstehungsgeschichte jeder Graphik mindestens im nachhinein als nach einem wohldefinierten Programm abgelaufen vorstellen, da auch im handwerklichen Vollzug Richtung und Gliederung zu erkennen und Etappen zu unterscheiden sind. Im Rahmen generativer Graphik jedoch ist das Programm mit seiner Struktur die erste und bewußt konstituierte Komponente des ästhetischen Erzeugungsprozesses, derart, daß von der Struktur der erzeugten Graphik jederzeit auf die dokumentierte Struktur des erzeugenden Programms zurückgefragt werden kann. Dabei ist die generative Graphik in der glücklichen Lage, in der sich die Astronomie erst nach der Erfindung des Teleskops befand. Sie kann nämlich die Programme, die sie erstellt, von den als Computern oder Datenverarbeitungsanlagen bezeichneten Maschinen auswerten lassen. Man hat den Computer als Komplexitätsfernrohr bezeichnet, weil er vorher unzugängliche Komplexität überhaupt erst auflösbar, dann allerdings sogar manipulierbar macht. Bild 33 diene als

Beispiel! Es ist — da Zeit kostbar ist — schlechterdings undenklich, ein derartiges Exemplar ästhetischer Information jemand von Hand nach vorgegebenem Programm erarbeiten zu lassen. Die Programme, als Erzeugungsprinzipien von graphischer Information, müssen in normierten Sprachen geschrieben sein, um von Datenverarbeitungsanlagen verstanden werden zu können. Daher ist Kapitel 2 der vorliegenden Arbeit dem Entwurf einer Programmiersprache für die generative Graphik gewidmet.

An einem sehr einfachen Beispiel zeigen wir nun, wie Gestaltung, Verteilung und Zusammenhang in die Realisation einer programmierten und maschinell erstellten Graphik eingehen können. Man betrachte die linke Spalte von Bild 5! Horizontale Strecken zufälliger, jedoch begrenzter Länge beginnen rechtsbündig und sind in gleichen Abständen vertikal übereinander angeordnet. Soll die Graphik von einer programmierbaren Maschine (einem Roboter) gezeichnet werden, so hat ein Programm für die säulenartige Gestaltung zu sorgen, es muß z. B. jede der Einzelstrecken an der gleichen vertikalen virtuellen Kante beginnen oder enden lassen. Das Programm wird vermutlich zyklische Struktur haben und vermittels sukzessiver Durchlaufungen eines Zyklus die Einzelstrecken individueller Länge realisieren, wobei jede Durchlaufung ein vertikales Abrücken der jeweils nächsten von der vorhergehenden Einzelstrecke einschließt — auf diese Weise ist der Zyklus im Programm das Schema der Verteilung der Einzelstrecken, die Verteilung der Einzelstrecken die sichtbare Spur der zyklischen Struktur des Programms. Den Strukturentsprechungen zwischen Programm und programmiert realisierter Information nachzugehen, wird eine der Hauptaufgaben unserer Untersuchung sein. Gestaltung

und Verteilung haben wir nun erwähnt, wie steht es aber mit dem Begriff des Zusammenhangs im Hinblick auf das Beispiel in Bild 5? Es fällt auf, daß das Bild vor allem durch die strenge, rasterartige Anordnung der Strecken zu einem Ganzen wird und dadurch Zusammenhang — Nexus — erhält; der Zusammenhang wird allerdings zwar von der Bildstruktur nahegelegt, jedoch erst durch eine Art Nacharbeit bei der Perzeption endgültig realisiert, denn nachweislich berühren sich die Einzelstrecken ja nicht, so daß man ebenso berechtigt wäre zu sagen, *das Bild zerfällt,* als *es hängt zusammen.* Die Perzeptionsabhängigkeit des Bildnexus, auf die wir hier stoßen, birgt schwierige Probleme, mit denen wir uns in Kapitel 3 befassen werden. Freilich ist die Perzeptionsabhängigkeit des Bildnexus nichts so Besonderes, wenn man bedenkt, daß die ästhetische Information als ein Zeichensystem ohnehin nur in der Zeichenfunktion, d. h. aber: in der Kommunikation, da ist. Das ist ja die durchgängige Thematik der Aesthetica. In der Kommunikation, d. h. im Hin- und Herspiel zwischen Zeichensender und Zeichenempfänger, Expedient und Perzipient, ist die ästhetische Information natürlich von beiden abhängig. Nur hat man sich in der Vergangenheit daran gewöhnt, Bilder als unveränderliche Objekte zu betrachten, und muß nun lernen, sich des Kommunikationscharakters der Bildbetrachtung mit der damit verbundenen wachen Reflexion auf Kommunikation bewußt zu werden. Dazu kann die generative Graphik insofern beitragen, als sie gezielt Exemplare von ästhetischer Information herzustellen gestattet, die Reflexion wachrufen.

Im obigen Zitat unterscheidet Bense *freie* und *gebundene* ästhetische Strukturen und kennzeichnet die ersten durch die Merkmale des Stochastischen und Intuitiven, die zweiten durch Festlegbarkeit im voraus und Deduk-

tion. Auch für die generative Graphik kann diese Unterscheidung getroffen werden, doch soll gesagt sein, daß wir den Bereich des Stochastischen, des Zufalls, als die eigentliche und fruchtbare Domäne der generativen Graphik betrachten — in Übereinstimmung mit den Aesthetica, in denen auch der Aufweis des wesentlich stochastischen Charakters der ästhetischen Information geführt wird. Im ersten Unterabschnitt von Kapitel 3 werden wir uns ausführlicher mit der These befassen, daß das stochastische Rauschen, das die Informationskanäle der Nachrichtentechniker so sehr stört, gerade das Interessante, das Informative, zur ästhetischen Information beisteuert. Schauen wir noch einmal kurz die beiden Teilbilder von Bild 5 an! Das linke Teilbild gehört zur ersten, das rechte zur zweiten der oben genannten beiden Klassen ästhetischer Information. Geht man im rechten Teilbild die Einzelstrecken von oben nach unten aufmerksam durch, so kann man bald die Streckenlängen aus dem bereits Wahrgenommenen deduzieren. Aus dem linken Teilbild ist diese Deduzibilität getilgt, deswegen ist das linke Teilbild reicher an Information als das rechte. In der Begriffswelt der Aesthetica heißt das informativ Neue in ästhetischer Information Innovation. Benutzen wir diesen Ausdruck, so können wir folgendermaßen ausdrücken, was wir mit generativer Graphik erreichen wollen: Sie zu einem Instrumentarium auszubauen, mit dessen Hilfe man Innovation in Gestalt graphischer Information sowohl hervorbringen als auch untersuchen kann. Genese und Analyse von Innovation ist der Zweck der generativen Graphik im Rahmen der allgemeinen generativen Ästhetik. In der Tat geht das Zitat aus AE, das wir im Augenblick besprechen und zu einer Interpretation der Thematik der generativen Graphik heranziehen, folgendermaßen weiter: „Da nun ästhetische Strukturen nur insofern *ästhetische Informa-*

tion enthalten, als sie Innovationen aufweisen und diese natürlich stets nur eine wahrscheinliche, keine definitive Wirklichkeit darstellen, kann man sagen, daß die künstliche Erzeugung von einer Norm abweichender Wahrscheinlichkeiten durch Theoreme und Programme das zentrale Motiv der generativen Ästhetik und ihrer Projekte ist." Mit wenigen Ausnahmen sind alle Bilder der vorliegenden Untersuchung daraufhin angelegt, unwahrscheinliche, von der Norm abweichende graphische Information hervorzubringen, und das Unwahrscheinliche, die Innovation in der einzelnen Graphik macht sie zum Ästhetikum. Material für die Sichtbarmachung der strukturellen Wirksamkeit von Innovation in ästhetischer Information vorzulegen, ist das Motiv für die Programmierung und maschinellen Erzeugung dieser Graphiken.

Kehren wir noch einmal zu dem eingangs zitierten Absatz aus den AE zurück! Bense sagt, daß sich die Genese ästhetischer Information dem Betrachter auf zwei Ebenen darbietet, einer makroästhetischen und einer mikroästhetischen. Als Ergebnis der Wirksamkeit der Prinzipien Gestaltung, Verteilung und Zusammenhang müssen an der generierten ästhetischen Information makroästhetisch Ordnung und Komplexität, mikroästhetisch Redundanz und Information wahrnehmbar werden. Es wird zu unserer Aufgabe gehören, die gegenseitige Abhängigkeit und das Ineinanderwirken der beiden Strukturebenen mit Hilfe der experimentellen Techniken zu erhellen, die vorher von uns zu entwickeln sind. Ordnung und Komplexität, wie sie anschaulich an den Bildern dieses Berichts abgelesen werden können, müssen verständlich werden, als determiniert durch Redundanz und Information — *Information* hier synonym zu *Innovation* gebraucht — und durch einen Prozeß, der

an vielen Stellen der AE unter der Bezeichnung *Dispersion* behandelt wird. *Innovation, Redundanz* und *Dispersion* erweisen sich dabei als die eigentlichen informationstheoretischen Fundamentalbegriffe der allgemeinen Informationsästhetik, wie insbesondere auch der generativen Graphik, denen man, bei vorsichtigem Gebrauch eines traditionsbeladenen Ausdrucks, den Rang von Kategorien zuerkennen kann. Die Bedeutung dieser Fundamentalbegriffe ergibt sich daraus, daß sie in vielen Fällen als statische oder dynamische Eigenschaften direkt mit wichtigen Strukturmerkmalen der Programme in Verbindung gebracht werden können, nach deren Direktive graphische Information entsteht. Schon am Beispiel des Programms zu Bild 5 ist ja gezeigt worden, was für nichttriviale Programme ganz allgemein gilt, daß nämlich die Programmstruktur zyklische Teilstrukturen enthält. Die Zyklen in Programmen legen sich dynamisch in die zeitlich wiederkehrenden Eigenschaften der Prozesse auseinander, die durch die Programme dirigiert werden. Die nun ebenfalls als zyklisch zu bezeichnenden Prozesse wiederum hinterlassen ihre Spur und in ihr deutlich Wiederkehrendes. Im Fall der generativen Graphik z. B. regiert das Programm den Griffel einer automatischen, programmsteuerbaren Zeichenmaschine. Was der Griffel aufzeichnet, ist die Spur des Prozesses, der dem Programm gehorcht. Alles Wiederkehrende in der aufgezeichneten Information ist Redundanz und die dynamische Setzung des Wiederkehrenden ist Dispersion. Das aber, was durch Dispersion vervielfacht wird und ursprünglich eines war, ist Innovation. In der Übersicht über Kapitel 3 (Abschnitt 1.3 dieser Einleitung), vollends in Kapitel 3 selbst, werden wir auseinanderlegen, was hier zunächst kategorisch hingeschrieben worden ist.

Es ist noch kurz zu erläutern, was die in den AE erwähn-

ten *metrischen, statistischen* und *topologischen* Bestimmungen ästhetischer Information (siehe wieder das Zitat am Eingang) für die generative Graphik bedeuten. Als metrisch sind z. B. alle informationsbestimmenden Zahlkonstanten anzusehen, die in Programme eingehen. Topologisches prägt sich sowohl in der Gestalt der Programme als auch in der gegenseitigen Lage ihrer Komponenten aus. Durch Statistik aber ist die generative Graphik durchgehend bestimmt, aufgrund der Verwendung von *Zufallsgeneratoren* in den Programmen, hierüber mehr in der nachfolgenden Übersicht über Kapitel 2. Metrische, statistische und topologische Bestimmungen von Programmen wiederum korrelieren mit entsprechenden Eigenschaften der programmiert generierten Information.

Obgleich wir nur einige wichtige Aussagen der AE über generative Ästhetik herangezogen haben, um darzutun, was generative Graphik bezweckt, benutzen die vorliegenden Untersuchungen doch den Begriffsapparat der Aesthetica in seiner ganzen Breite. Wo bestimmte Stellen der AE direkt einschlägig waren, wurden sie im einzelnen zitiert. Einige andere Literaturstellen sind in Fußnoten oder im Text angeführt. Ein ausführliches Literaturverzeichnis befindet sich am Schluß.

1.2. Übersicht über Kapitel 2

Mit der Schaffung der generativen Ästhetik geht die Informationsästhetik ins Laboratorium und bedient sich technischer Experimentiereinrichtungen. Das wichtigste Instrument, das sie dabei benutzt, ist die Datenverarbeitungsanlage, auch Computer genannt. Wie schon im vorhergehenden Abschnitt erwähnt wurde, ist die Daten-

verarbeitungsanlage ein Werkzeug zur Bewältigung von komplexer Information. Dieses Werkzeug hat sich als derart effektiv erwiesen, daß es die Geschichte der verschiedensten Wissenschaften tiefgehend und intensiv beeinflußt hat. Nicht nur die Naturwissenschaften sind betroffen, sondern auch Wissenszweige, die sich vorher der Behandlung mit Hilfe maschinell abwickelbarer Prozeduren fast ganz verschlossen hatten, wie z. B. die Literaturwissenschaften. Es ist wohlberechtigt, zu behaupten, daß die Gestalt der Wissenschaften von einem singulären Punkt in den späten vierziger Jahren ab anders ist, als vorher, der singuläre Punkt bezeichnet das Auftauchen des Computers. Wissenschaften, die nach diesem Zeitpunkt entstanden, gerieten von vornherein in fruchtbare Abhängigkeit von der maschinellen Datenverarbeitung, zu diesen Wissenschaften zählt auch die generative Graphik. Man bedenke dabei, daß die vorliegenden Untersuchungen die Problematik der generativen Graphik nur an einigen wenigen Stellen aufweisen können und daß die angebotenen Lösungen der Weiterführung und Verfeinerung bedürfen. Wir kommen darauf im Abschnitt 1.4 der Einleitung zurück, der Problemlinien aufzeigt, die wir des voraussehbaren umfangreichen Problemlösungsaufwandes wegen nicht verfolgen konnten. Auf einen Nenner gebracht, ist es die Synthetisierung und Analysierung des Bildmaterials der generativen Graphik, die ihrer Komplexität wegen nur mit Hilfe der Datenverarbeitungsanlagen befriedigend abgewickelt werden kann.

Im Fall der maschinellen Synthese graphisch-ästhetischer Information hat die verwendete Datenverarbeitungsanlage zwei Bedingungen zu erfüllen: Sie muß das Programm, nach dessen Anweisungen die Graphik synthetisiert werden soll, vor der Synthese in den zum

Computer gehörigen Datenspeicher (das Gedächtnis) aufgenommen haben, und sie muß prinzipiell fähig sein, eine automatische Zeichenmaschine zu steuern, um die bei der Abwicklung des Syntheseprogramms anfallenden Daten in sichtbare graphische Zeichen umsetzen zu können. Oft wird die Umsetzung der elektrischen Steuerimpulse der Datenverarbeitungsanlage in graphische Zeichen derart in zwei Schritten vorgenommen, daß die Datenverarbeitungsanlage einen Datenträger — meistens einen gelochten Papierstreifen — herstellt, der dann von der automatischen Zeichenmaschine gelesen und zu direkt sichtbarer graphischer Information weiterverarbeitet wird. Der Zwischenschritt der Herstellung eines Datenträgers ist jedoch methodologisch irrelevant, so daß es erlaubt ist, die Zeichenmaschine als unmittelbare, effizierende Peripherie des Computers aufzufassen. Damit reduziert sich, soweit die Datenverarbeitungsanlage mit ihrem Zeicheneffektor als funktionierend vorausgesetzt wird, das Syntheseproblem auf das Programmierungsproblem. Wer graphische Information haben will, muß es dem automatischen Computer-Graphiker in geeigneter Form mitteilen, wobei diese Form syntaktische und semantische Form des Programms ist. Kapitel 2 des vorliegenden Berichts ist nun nichts anderes, als ein Programmierkurs, d. h. jedoch, es ist ein Sprachkurs, da jede Klasse von Programmen gewissen grammatikalischen, d. h. sprachlichen Regeln entsprechen muß. Der Weg, den Kapitel 2 bei seiner Sprachbelehrung einschlägt, ist die Definition einer Sprache für die generative Graphik als der geeigneten Erweiterung einer in der wissenschaftlichen Programmierung hochbewährten Sprache um generativ-graphische Ausdrucksmittel. Die Sprache G, die auf diese Weise von Kapitel 2 gelehrt wird, ist ein typischer Fall derjenigen Sprachform, die in der Programmiertheorie als *Pro-*

blemorientierte Sprache bezeichnet wird. Solche problemorientierten Sprachen sind in jüngster Vergangenheit für viele Wissenschaftszweige (so z. B. die technische Statik) geschaffen worden. Die Leistung des Kapitels 2 ist also die Bereitstellung der problemorientierten Sprache G für die generative Graphik.

In dieser Übersicht seien nur zwei grundlegende Anweisungen erwähnt, die in der Sprache G vorkommen. Die erste ist LINE. Sind P und Q die Koordinaten eines Punktes in einem kartesischen Achsenkreuz, so genügt die Mitteilung der sprachlichen Anweisung LINE(P,Q) an den Computer, den Zeichengriffel eine gerade Linie von seinem augenblicklichen Ort zum Punkt P,Q ziehen zu lassen. Die zweite grundlegende Anweisung LEER befördert, wenn sie z. B. nach der Anweisung LINE(P,Q) in der speziellen Form LEER(S,T) an den Computer weitergegeben wird, den Griffel von seinem augenblicklichen Ort P,Q zum neuen Punkt S,T — jedoch so, daß der Griffel keine sichtbare Spur auf dem Zeichenpapier hinterläßt. Es dürfte sofort einleuchten, daß alle Bilder dieses Berichts, die aus Geradenstücken zusammengesetzt sind, durch geduldige Aneinanderreihung von LINE- und LEER-Anweisungen programmiert werden können. An dieser Stelle jedoch wird die Einbettung der Sprache G in ihre Trägersprache — es ist dies die berühmte Programmiersprache ALGOL — wirksam. Die Sprache ALGOL nämlich erlaubt durch den Aufbau von Programmzyklen die drastische Raffung der strukturlosen Aneinanderreihung von LINE- und LEER-Anweisungen. Wie man Programmzyklen konstruiert und wie man überdies noch logische Steuerbedingungen in die so errichteten Programmstrukturen einfügt, ferner, durch welche Sprachmittel man Programme hierarchisch gliedert, behandelt Abschnitt 2.1.

Die sprachlichen Mittel, die in Abschnitt 2.1 erläutert werden, reichen zur Programmierung der Genese rein deterministisch bestimmter Bildinformation aus. Typische Beispiele für den damit überstrichenen Informationsbereich zeigen die Bilder 1 bis 3. Nun ist das Kunstschöne aber durchgängig durch Unregelmäßigkeit ausgezeichnet, d. h. dadurch, daß bestimmte Zeichen immer nur mit einer gewissen Wahrscheinlichkeit an bestimmten Stellen in Zeichengefügen angetroffen werden. Aus diesem Grund sind Bilder der Art 1 bis 3 für die generative Graphik ziemlich unergiebig, insofern diese letzten Endes auf die experimentelle Erfassung und simulative Modellierung des Gesamtbereichs graphischer ästhetischer Information aus ist. Da die normalerweise zur Verfügung stehenden Datenverarbeitungsanlagen keine echten Zufallsquellen (wie sie z. B. durch Ausnützung des Zerfalls radioaktiver Substanzen realisiert werden könnten) enthalten, die der Programmierer unmittelbar als Informationsquellen verwenden könnte, benutzt man zahlentheoretische Eigenschaften der natürlichen Zahlenreihe, um Serien von Zufallsgrößen arithmetisch zu erzeugen. Dies gelingt völlig befriedigend, wie die Bilder ab Bild 12 (mit Ausnahme von Bild 43) anschaulich zeigen. Die Bilder 6 bis 12 führen den Übergang von *entarteten* bis zu *vollkommenen* arithmetischen Zufallsquellen vor. Wir bezeichnen die arithmetischen Zufallsquellen als Pseudozufallsgeneratoren, meistens jedoch kurz als Zufallsgeneratoren.

In Abschnitt 2.2 werden die Zufallsgeneratoren so in die Sprache G eingefügt, daß sie in bequemster Weise gehandhabt werden können. Sechs voneinander unabhängige Zufallsgeneratoren werden bereitgestellt, die einfach durch Angabe der Namen J1 bis J6 aufgerufen und aktiviert werden. So befiehlt z. B. die Anweisung LI-

NE(J1,J2) der Datenverarbeitungsanlage, den Zeichengriffel auf einer geraden Linie von seinem augenblicklichen Ort zum Punkt mit den zufälligen kartesischen Koordinaten J1,J2 zu bewegen. Vor dem ersten Aufruf eines Zufallsgenerators kann ein beliebiges Zahlenintervall ausgewählt werden, in das der Zufallsgenerator die von ihm erzeugten Zahlenwerte hineinstreut.

Das geometrische Arsenal der Sprache G besteht zunächst aus beliebig in der Zeichenfläche gelagerten Strecken. Andere geometrische Elemente, etwa Kreise, Ellipsen usw. könnten hinzugefügt werden. Wir beschränken uns auf die Erweiterung der Sprache G um Kreisbögen und Rechteckflächen und erklären im abschließenden Abschnitt 2.3, wie man mit diesen neuen geometrischen Elementen programmierend umgeht. Beispiele zu Abschnitt 2.3 zeigen die Bilder 13 bis 15 und Bild 17.

Die Sprache G ist insoweit allgemeinverwendbar, als für eine bestimmte Datenverarbeitungsanlage, die zu graphischen Experimenten benutzt werden soll, ein sogenannter ALGOL-Übersetzer zur Verfügung steht. Der ALGOL-Übersetzer ist eine Art Rüstprogramm, das die Datenverarbeitungsanlage in den Stand setzt, in der Programmiersprache ALGOL geschriebene Programme zu rezipieren und zu verarbeiten. Für eine große Anzahl bedeutender Datenverarbeitungsanlagen existieren jedoch ALGOL-Übersetzer.

Wie wir schon erwähnten, ist die Sprache G für die generative Graphik eine Erweiterungssprache der Sprache ALGOL. Was die Sprachelemente OPEN, CLOSE, LEER, LINE und das in Abschnitt 2.3.1 behandelte ZIRK anbelangt, fußt G auf einer unterlagerten Erweiterungs-

sprache von ALGOL, die von Herrn Horst Glasauer von der SIEMENS AG entwickelt wurde. Das Glasauersche System kennt u. a. die Elemente OPEN, CLOSE, TAKE, CIRC und SONK. Der zuletzt genannte Begriff bedeutet *Sonderkommando,* wir werden das Sonderkommando SONK(11) in Abschnitt 3.2.1 benutzen. Bestimmte Sonderkommandos dienen zum Heben und Senken des Zeichengriffels. Diese Sonderkommandos sind zum Aufbau unserer Anweisungen LEER, LINE und ZIRK derart herangezogen worden, daß jetzt das Heben und Senken des Griffels automatisch abläuft. LEER, LINE und ZIRK halten ferner die überall in den Programmen zugänglichen Größen X und Y, die den Momentanort des Griffels bedeuten, auf dem neuesten Stand. Im Fall des vorliegenden Berichts wurde als technische Ausrüstung eine Datenverarbeitungsanlage des Hauses SIEMENS und ein automatischer Zeichentisch ZUSE GRAPHOMAT benutzt. Ein besonderes Verdienst des Glasauerschen Systems besteht darin, die Genese der Elementarbefehle für den GRAPHOMAT zu leisten und diese Funktion zum ALGOL-System der Siemens-Datenverarbeitungsanlage zu addieren.

1.3. Übersicht über Kapitel 3

Zu Beginn des Kapitels 3 liegen die Mittel zur Herstellung graphischer Information bereit und es bestünde nun die Möglichkeit, diese Mittel direkt zur Herstellung von Kunstwerken zu benutzen, so, wie Pinsel und Palette oder die photographische Kamera schon lange dem Künstler dienten. Wir betrachten eine solche Möglichkeit des Einsatzes der Datenverarbeitungsanlagen als durchaus legal und vielversprechend, es ist jedoch nicht unsere Aufgabe, sie weiter zu verfolgen. Generative Ästhetik wird ja erst dadurch zu einem fruchtbringenden Teilge-

biet der Informationsästhetik, daß sie die ästhetische Reflexion auf die von ihr hervorgebrachte Information in Gang setzt und vorantreibt. Im Fall der generativen Graphik braucht das Hauptproblem nicht erst sorgfältig herauspräpariert zu werden, sondern liegt als die Frage nach den Strukturbeziehungen zwischen generierendem Programm und generiertem Bild offen da. Beide, Bild und Programm sind Informationen, das Programm texthafte, das Bild bildhafte. Im Sinn dieser Bemerkung ist das Hauptproblem der generativen Graphik die Frage nach den Strukturbeziehungen zweier spezifischer Informationssysteme.

Genetisch betrachtet, scheint das Problem ein rein ästhetisch-kosmogonisches zu sein: Demiurgisch speit der Computer ästhetische Information in die Welt und die Welt absorbiert die in sie einströmende Information, wie sie ankommt. Die AE gehen ja an den verschiedensten Stellen auf kosmologische und kosmogonische Grundfragen der Ästhetik ein, z. B. in dem Abschnitt „Ästhetische Kommunikation und Information" (AE, Seite 208). So heißt es AE, Seite 209: „Aus der Geschichte der Ästhetik und der Kunsttheorie läßt sich belegen, daß das Wesentliche der ästhetischen Prozesse zu fast allen Zeiten in einem eigentümlichen Ordnungsvorgang gesehen wurde, dessen Determination in größtem Ausmaße von der freien Wahl des Künstlers abhängig gemacht werden sollte. Wie in unserer Auffassung bezog man mehr oder weniger deutlich Verteilung und Wahl sowohl auf inhaltliche wie auch formale Bestimmungsstücke. Sowohl das, was unter Verteilung wie auch das, was unter Wahl verstanden wird, hat Zeichencharakter. Information und Kommunikation, aufgefaßt als Darstellung und als Darstellung einer Darstellung, sind letztlich Zeichencharaktere, verständlich und meßbar als bestimmte Verteilung

oder Anordnung gewisser Materialien, hervorgegangen aus Auswahlen." Spezialisiert man das hier Gesagte auf graphische Zeichensysteme und macht man sich klar, daß auch der Computer beim Ablauf eines generativen Programms letztlich nichts anderes macht, als Zeichen aus einem gewissen Arsenal auszuwählen und zu verteilen, so folgt, daß der Computer nichts anderes ist, als ein besonders exakt analysierbares Modell des künstlerischen Erzeugungsprozesses. Tatsächlich ist das Hauptziel unserer Untersuchung die Auslegung dieser Erzeugungsthematik. Angesichts recht auffälliger Merkmale vieler graphischen Informationen kommt die generative Graphik jedoch in eine schwierige Lage, wenn sie sich nur um die Untersuchung des Informationserzeugungsprozesses kümmert. Manche Struktureigenschaften generierter Information lassen sich nämlich nicht auf Struktureigenschaften des generierenden Automaten zurückführen. Ein Beispiel zeigt Bild 17, dessen unteres Teilbild sich vom oberen strukturell nur dadurch unterscheidet, daß der Schwankungsbereich der Radien für die Einzelkreise verändert wurde. An der Struktur des erzeugenden Programms ändert sich durch diese Maßnahme überhaupt nichts, das Produkt ändert sich jedoch in krasser Weise. Freilich ist es eine dem makroästhetischen Bereich angehörende Vorentscheidung des Programmierers, die Kreisradien zu ändern, wesentlich ist jedoch, daß diese Entscheidung das ganze komplizierte und durchaus mikroästhetische Überschneidungssystem der Kreislinien in die ästhetische Welt setzt. Es kommt der wahrscheinlich wohlbegründbare Verdacht hinzu, daß verschiedene Betrachter, abhängig von einer gewissen individuellen *ästhetischen Sensibilität* verschieden komplexe Überschneidungssysteme perzipieren.

Die methodologische Folge für eine modellierend vorge-

hende Ästhetik ist die, daß es der volle Kommunikationsprozeß eigentlich mit drei Maschinen zu tun hat, nicht nur mit dem einen Erzeugungsautomaten. Die erste Maschine generiert Information, die zweite ist ein Informationskanal und transportiert sie, die dritte ist ein Informationsanalysator und rezipiert oder perzipiert die ästhetische Information. Freilich ist es möglich, den ästhetischen Kommunikationsprozeß als einen restlos geschlossenen, ohne erkennbare Trennstellen, anzusehen, dementsprechend die Welt als einen geschlossenen ästhetischen Kosmos. Diese Betrachtungsweise würde jedoch so drängende Fragen, z. B. ob und in welcher Weise die informationserzeugende Instanz der informationsaufnehmenden kreativ-potentiell *voraus* ist, schwer modellmäßig beantwortbar machen. Max Bense trägt dieser Problematik Rechnung, indem er ausdrücklich auf das komplizierte System analytischer und interpretativer Prozesse hinweist, die auf der Perzeptionsseite der ästhetischen Kommunikationskette Platz greifen. Er sagt (AE, Seite 210): „Wenn eine ästhetische Produktion, z. B. ein Kunstwerk, als Darstellung im Sinne einer ästhetischen Information (also als Information über Verteilung und Auswahl ästhetischer Zeichen und Strukturen) verstanden wird, dann sind Analyse, Interpretation und Kritik bereits Formen der Replikation dieser Darstellung, somit Formen der Kommunikation, reproduzierte Darstellung der ursprünglich produzierten in einem neuen Darstellungsfeld. In dem Maße wie die Kommunikation neue Mittel und deshalb neue Zeichen benutzt, ist sie im Verhältnis zur Information selektiv. Der Selektionsprozeß, mit dem der Prozeß der ästhetischen Produktion anhebt, erzeugt zuerst die ästhetische Information und setzt sich fort in allen Formen ästhetischer Kommunikation; es tritt eine Folge von Darstellungsprozessen ein, beginnend mit der Darstellung, die sich

fortsetzt in eine Darstellung der Darstellung usw." Besonders der Terminus „reproduzierte Darstellung der ursprünglich produzierten in einem neuen Darstellungsfeld" ist an dieser Stelle für uns wichtig. Wir müssen nämlich schon so *frühe* Prozesse der Darstellung, wie die perzipientenseitige Erfassung von Zeichenüberlagerungen (man betrachte wieder Bild 17), als reflexive Reproduktionen in einem neuen Darstellungsfeld, nämlich dem perzeptorischen, auffassen. Diese Problematik wird sich wie ein roter Faden durch das ganze Kapitel 3 ziehen und in Abschnitt 3.3 über Gestalten unwiderruflich feste Form annehmen.

Die Problematik zusammenfassend, ist zu sagen, daß die generative Graphik sich ständig selbst übersteigt in eine volle kommunikative Graphik hinein. Das Überschneidungssystem der Kreise in Bild 17 ist eben nicht aus dem genetischen Algorithmus von Bild 17 deduzierbar, sondern es evolviert aus der Struktur des Perzipienten. Die generative Graphik liefert also nur die eine Hälfte der ästhetischen Wahrheit. Diese Erkenntnis verursacht den eigentümlich labilen Status des Kapitels 3. Ein nach unserer Meinung so fundamentaler Begriff wie *endogene Redundanz* (siehe Abschnitt 3.2.6) kann nur diskutiert werden, wenn die Grenzen der generativen Graphik bewußt und methodisch überschritten werden. Das Thema der vollen kommunikativen Graphik jedoch ist eine große Sache, zu groß für unseren Rahmen, etwas für *Large-Scale-Research,* wie wir im Abschnitt 1.4 dieser Einleitung verständlich zu machen hoffen.

Die Beziehungen zwischen Programm und Bild, ihre gegenseitige komplexe Strukturzuordnung, bestimmen jedenfalls das Thema des Kapitels 3. In den Hauptabschnitten dieses Kapitels werden die Produkte der gene-

rativen Graphik, die insgesamt Modelle von Bildinformation überhaupt sind, aus zwei Aspekten betrachtet: Bestimmtheit durch gewisse ästhetische Grundgegebenheiten einerseits, Zugehörigkeit zu gewissen Informationsklassen andererseits. Abschnitt 3.1 ist dem ersten Aspekt gewidmet, die Abschnitte 3.2 und 3.3 befassen sich mit dem Klassifikationsproblem. Schon durch die Überschrift von Abschnitt 3.3: „Gestalt und Gegenstand" wird angedeutet, daß die Gestalten eine Bildinformationsklasse höchster Bedeutung darstellen.

Einiges zum Abschnitt „Ästhetische Kategorien"

Abschnitt 3.1 beginnt zunächst mit einer nochmaligen Klärung und Deutung des Begriffs des bildgenerierenden Automaten. Wie bereits gesagt wurde, ist der bildgenerierende Automat eine Datenverarbeitungsanlage, die befähigt ist, Information an eine steuerbare Zeichenmaschine zu liefern. Bevor eine solche Datenverarbeitungsanlage jedoch ein Bild erzeugen kann, muß sie ein Programm zur Genese des Bildes in ihr Gedächtnis einlesen. Eine Datenverarbeitungsanlage, die auf diese Weise mit einem speziellen Programm zur Erzeugung eines speziellen Bildes *geladen* wird, kann als äquivalent mit einem speziellen Automaten angesehen werden, der nur und nichts anderes als dieses Bild generieren kann. Dieser Schluß enthüllt auch eine Äquivalenz zwischen Programmen und speziellen Automaten. Programme und spezielle Automaten bezeichnet deshalb das Kapitel 3 zusammenfassend als Bildgeneratoren, kurz als Generatoren. Es erscheint zunächst als widerspruchsvoll, eine Äquivalenz herzustellen zwischen einem Programm als einem Text einerseits, einem Automaten als einem Gerät andererseits. Doch sind Programm bzw. spezieller

Automat nichts anderes als der prozeßhaft-funktionelle bzw. der materiell-strukturelle Aspekt der gleichen Informationserzeugungsmöglichkeit.

In vielen Fällen hängen Programme noch von Größen ab, die als Parameter bezeichnet werden. Eine spezielle Parameterwahl erzeugt eine bestimmte Bildvariante. Geht man jetzt gedanklich wieder zum Begriff des speziellen Automaten über, so erweist sich ein jedes *parametrisierbare* Programm als äquivalent mit einem speziellen Automaten, der Eingangsinformation (die Parameterwerte) aufnimmt, Ausgangsinformation (das generierte Bild) liefert. Automaten mit Eingangs- und Ausgangsinformation werden in der Informationstheorie allgemein als *black boxes* oder auch als Transduktoren bezeichnet.

Programme, spezielle Automaten, Generatoren, sind also untereinander äquivalente Begriffe. Ein Transduktor wiederum verwandelt sich in einen Generator, wenn seine zunächst frei wählbare Eingangsinformation feste Werte annimmt.

Den wichtigen Begriffen, die damit eingeführt worden sind, fügt Abschnitt 3.1 eingangs noch eine Erläuterung der *ästhetischen Kategorien* Innovation, Redundanz und Dispersion hinzu. Der Begriff *Kategorie* ist hier jedoch wirklich streng regional auf das Gebiet der Informationsästhetik beschränkt, ein ästhetischer Binnenbegriff. Innovation ist ästhetische Information, deren Einzigartigkeit durch Redundanz eingeschränkt, dadurch aber erst perzipierbar wird. Dispersion verteilt Innovation, läßt sie in das Bildgefüge eingehen und dadurch sichtbar werden.

Abschnitt 3.1.1 demonstriert, daß in einfach gelagerten

Fällen die Parameter, von denen die Struktur einer Bildinformation abhängt, in selektorische und kompositorische aufgeteilt werden können. Z. B. erweist sich derjenige Parameter, der für das Typische der Einzelfiguren in den Bildern 15 und 16 verantwortlich ist, als selektorisch, jedoch nicht, weil er das Typische selektiert, sondern weil er mit Hilfe von Zufallsgeneratoren die Selektion der Individualität der Einzelfiguren bewirkt. Die Selektion des Typischen selbst ist eine ästhetische Vorentscheidung des Programmierers, auch diese wichtige Fragestellung wird hier erläutert. Kompositorisch wirken die zeichenverteilenden Parameter. Allerdings ist die Zerfällung in selektorische und kompositorische Parameter, damit in die Prozesse Zeichenauswahl und Zeichenverteilung, nur bei bestimmten, einfach aufgebauten Bildern möglich. Im wesentlichen sind diese Bilder Kompositionen, deren Einzelzeichen perzeptorisch völlig separierbar sind. Diese Betrachtungen werden durch Abschnitt 3.1.2 in dem Sinn ergänzt, daß gezeigt wird, wie die Sortierung der Eingangsparameter eines Transduktors in selektorische und kompositorische einer bestimmten Zerlegung des Transduktors in Teiltransduktoren entspricht.

In Abschnitt 3.1.3 wird das schon öfter erwähnte Bild 17 behandelt, das eng mit dem Problem der ästhetischen Perzeptivität zusammenhängt. Man wird zustimmen, wenn das Kreisüberschneidungssystem in der unteren Hälfte von Bild 17 als kompositorisch wirksam bezeichnet wird. Nun hat allerdings schon Abschnitt 3.1.1 den Kreisradius in Bild 17, dessen Schwankungsbreite sich entscheidend auf die Bildstruktur auswirkt, als selektorisch klassifiziert. Damit ist gezeigt, daß selektorische Parameter komponierend wirken können, daß die strenge Zweiteilung der Parameter also nicht immer sinnvoll

ist. Der kritische Fall tritt genau dann ein, wenn Zeichenkomplexe sich im Bild stark überlagern, genau dann also, wenn die perzeptorische Aktivität aufgerufen werden muß, sich an der Vereindeutigung des Bildgefüges zu beteiligen (z. B. ist es nicht möglich, die Halbmonde in der unteren Hälfte von Bild 17 aus der Struktur des Bildgenerators herzuleiten).

Bis an diese Stelle hat der Inhalt des zweiten Kapitels vorbereitenden Charakter. Mit Abschnitt 3.1.4 jedoch erreichen wir Kerngebiet. Wir beginnen, die Rolle des Zufalls bei der Bildentstehung zu untersuchen. Der Zufall wird dadurch methodisch genützt, daß Zufallsgeneratoren an der Bilderzeugung beteiligt werden. Zwar werden die bilderzeugenden Transduktoren absichtsvoll programmiert und ihre Eingangsinformation resultiert aus einer Vorentscheidung. Verwendet der Programmierer jedoch Zufallsgeneratoren, so vermindert sich sein Einfluß auf die generierte Bildstruktur. Aus der Erfahrung im Umgang mit Zufallsgeneratoren darf mitgeteilt werden, daß sie dem Programmierer immer neue Überraschungen bereiten. Die Funktion der Zufallsgeneratoren rennt in der ästhetischen Programmierpraxis so oft alle Planungen über den Haufen, daß man sie ohne jede Übertreibung den zweiten schöpferischen Faktor neben der Intention des Programmierers nennen darf. Wir entsinnen uns, daß Innovation als das wesentlich Neuartige an ästhetischer Information bezeichnet wurde. Im Hinblick auf diesen Begriff verknüpft Abschnitt 3.1.4 die beiden schöpferischen Faktoren der Bildgenese mit den beiden Begriffen *Makroinnovation* und *Mikroinnovation.* Innovation wird damit zu einer Summe aus zwei Anteilen, jedoch auch hier hält die Eigengesetzlichkeit der generativen Ästhetik (die wieder einmal dabei ist, sich selbst zu transzendieren — oder sich zu iterieren,

wie man will!) eine Überraschung bereit: Makroinnovation, als der schöpferische, im Makrobereich sich vollziehende Entwurf des Programmierers, gibt ja die großräumigen Umrisse, Schwerpunkte und weitgespannten Brücken des Bildgefüges. Großräumiges resultiert jedoch auch unerwartet aus der Funktion der Zufallsgeneratoren — ein ganz merkwürdiges Beispiel liefert das dem aufmerksamen Beobachter sichtbare Andreaskreuz innerhalb der Ellipse in Bild 52, andere Beispiele sind die aus den Bildern 49 bis 51 herauslösbaren Teilgestalten. Diese sekundäre, großräumige Information bezeichnen wir als inhärente oder endogene Makroinnovation, wobei wir nun schon dem Inhalt späterer Abschnitte vorausgegriffen haben. Den Gegensatz zur inhärenten Makroinnovation bildet die oben eingeführte, vom Programmierer herrührende Makroinnovation, die jetzt genauer als induziert oder exogen zu bezeichnen ist. Mikroinnovation und endogene Makroinnovation gehören freilich der von Max Bense beschriebenen Mikroästhetik an, dementsprechend schließt Abschnitt 3.1.4 mit der Bemerkung: „Mikroästhetik ist in diesem Sinn die Theorie der Generatoren im Hinblick auf ihre von den Zufallsgeneratoren abhängigen Eigenschaften." Das Problem der ästhetischen Vorentscheidung löst Abschnitt 3.1.4 für den Bereich der generativen Graphik, indem er die ästhetische Vorentscheidung als identisch mit der Konstituierung der induzierten Makroinnovation erkennen läßt.

Abschnitt 3.1.5 formuliert eine summarische Lösung des Grundproblems der generativen Graphik, das nach dem Strukturzusammenhang zwischen Generator und Produkt fragt: „Die mikroinnovativen Anteile am Produkt hängen von der Aktivität der Zufallsgeneratoren des Generators, die (exogen) redundanten von seiner Wei-

chen- und Schleifenstruktur ab, insoweit diese nichts mit der Maschinerie der Zufallsgeneratoren zu tun hat." Hierbei sind *Weiche* und *Schleife* immer wiederkehrende Strukturmerkmale von Programmen. Die Redundanz wird hier als Resultante der Programmgliederung angesehen. Daraus folgt der enge Zusammenhang zwischen Redundanz, Dispersion und induzierter Makroinnovation. Um ihn klar herauszuarbeiten, führen wir noch einmal die Schritte auf, die bei der maschinellen Genese graphischer Information durchlaufen werden: Am Anfang steht die ästhetische Intention des Programmierers, die jedoch zwei voneinander abgesetzte Komponenten enthält. Die erste Komponente schlägt sich beim Programmieren in der großräumigen Weichen- und Schleifenstruktur des Programms nieder. Sie wird in die Programminformation induziert, wird Teil der Programminformation. Diese Komponente ist ihrer Natur nach Innovation und wirkt sich durch die Programmstruktur hindurch auch im generierten Bild aus. Die Auswirkung vollzieht sich im nächsten Schritt der Genese, der darin besteht, daß das Programm maschinell ausgewertet wird. Dabei entfaltet sich der Dispersionsprozeß, der die genannte erste Komponente als induzierte Makroinnovation zutage treten läßt, jedoch multipel, der Multiplizität der Schleifendurchlaufung gehorchend. Die zweite Komponente der ästhetischen Intention ist passiv, sie läßt Raum für die mikroinnovative Wirksamkeit der Zufallsgeneratoren.

Originelles, Innovation, findet sich demgemäß immer als Redundanz in der Graphik. Man betrachte z. B. die Bilder 1 und 4! Originelles in ihnen ist die Falterfigur, sie aber tritt multipel und damit redundant auf. Man betrachte mit der gleichen Überlegung die Bilder 2, 3, 5 und andere.

Abschnitt 3.1.7 bringt die mit Zufallsgeneratoren versehenen Transduktoren in Zusammenhang mit den verrauschten, d. h. mit Störinformation beaufschlagten Kanälen der Nachrichtentheorie. Es zeigt sich, daß die additive Störinformation mit der in der generativen Ästhetik durchaus erwünschten Mikroinnovation übereinstimmt. An dieser Stelle sei eine Rechtfertigung dafür vorgebracht, daß wir den Begriff der Entropie, den MAX BENSE auch innerhalb der Informationsästhetik benutzt, nicht verwenden. Die Eigenschaften von Informationskanälen werden ja mit Hilfe des Entropiebegriffs erklärt und insbesondere zahlenmäßig festgelegt. Weil wir hier ausschließlich nichtnumerische Ästhetik treiben, wenn man vom notwendigen Umgang mit Zahlenwerten innerhalb von Programmen absieht, haben wir auch das Entropiemaß vermieden. Durchaus mit dem begrifflichen und praktischen Gebrauch der AE im Einklang befindlich, ist jedoch unser Begriff der Mikroinnovation der Sache nach Entropie.

Wir schließen die Übersicht über Abschnitt 3.1 mit einigen *Thesen* ab, die man als die Eigenleistung dieses Teils unserer Untersuchungen bezeichnen kann.

1. Programme steuern Zeichenautomaten und generieren dadurch graphisch-ästhetische Information. Gegenstand der generativen Graphik ist die Erarbeitung von Programmen und die Untersuchung der Beziehungen zwischen ihnen und der nach ihrer Vorschrift erzeugten Information.

2. Der Computer ist ein relevantes und unabdingbares Werkzeug der generativen Graphik, weil er die Voraussetzung für die Aktualisierung hinreichend interessanter und deshalb komplizierter Programme ist.

Der Computer ist aus der generativen Graphik ebensowenig eliminierbar, wie das Fernrohr aus der Astronomie.

3. Das immer gegenwärtige Zufallselement im künstlerischen Erzeugungsprozeß erfaßt die generative Graphik modellmäßig durch den Einbau von Zufallsgeneratoren in die Programme.

4. Die Zufallsgeneratoren beteiligen sich an der Strukturierung der generierten Information, schaffen dabei unvorhersehbar Neues und erweisen sich dadurch als die zweite schöpferische Instanz neben dem Programmierer, der die globale Programmstruktur und projektiv durch sie hindurch die großräumige Struktur der zu generierenden Information entwirft. Da die Zufallsgeneratoren der Intention des Programmierers entgegenwirken können, wenn er ihnen auch im allgemeinen mit Absicht Wirkungsfreiheit einräumen wird, dürfen sie als die Antagonisten der Programmiertätigkeit bezeichnet werden.

5. Programmierung bzw. Zufallsgenese kennzeichnen den makro- bzw. mikroästhetischen Bereich innerhalb der generativen Graphik. Die Innovationen, die in den beiden Bereichen entspringen und induzierte Makro- bzw. endogene Mikroinnovationen heißen, sind voneinander unabhängige Komponenten ästhetischer Information. In der generativen Graphik ist ästhetische Vorentscheidung das gleiche wie die Tätigkeit des Programmierers, auch das gleiche wie die Erzeugung induzierter Makroinnovation.

6. Überlagerung in der Fläche von zeitlich nacheinander generierten Teilen einer graphischen Zeichenkonstel-

lation kommt häufig vor und ist nichts Besonderes. Durch die Tätigkeit der Zufallsgeneratoren kann die Überlagerung jedoch extrem komplizierte innovative Zeichenverhältnisse und Zeichenzusammenhänge schaffen, die perzeptorisch mehrdeutig sind und deren Vereindeutigung weder immer generativ, d. h. rein vom Programm her, erzwungen werden kann, noch auch immer erstrebenswert ist. Wir deuten diese Erscheinung als Selbsttranszendierung der generativen Graphik in eine volle kommunikative Graphik hinein, die modellmäßig den vollständigen Informationskanal, einschließlich Generator und Perzeptor, zu untersuchen hat. Die Durchführung dieses Vorhabens steht noch aus und bedingt *Large-Scale-Forschung.*

7. Die unter Punkt 6 erwähnten, durch den *Überlagerungseffekt* bedingten Zeicheneffekte äußern sich nicht nur mikroästhetisch punktuell oder lokal, sondern beeinflussen auch die großräumige Bildstruktur. Dieser Einfluß ist seinem Charakter nach Makroinnovation, jedoch nicht induzierte, sondern ihrer Herkunft nach als endogen zu bezeichnende. Daß die Perzeptoraktivität sich an der Endogenität der endogenen Makroinnovation beteiligt, wird nicht bezweifelt.

Wir behaupten natürlich nicht, daß die sieben *Thesen* eine auch nur grobe Übersicht über Ergebnisse und insbesondere Ergiebigkeit der generativen Graphik vermitteln können, der generativen Graphik als eines Gebietes, das über dieses Buch hinausgeht.

Einiges zum Abschnitt „Gradationen"

Wenn wir uns auch in den vorliegenden Studien nicht mit numerischer Ästhetik beschäftigen[1]), die es auf Vermessung ästhetischer Information abgesehen hat, so wollen wir doch einer späteren Ergänzung der generativen Graphik durch numerische Analysen keine Hindernisse in den Weg legen. Vielmehr versuchen wir schon bei der Behandlung des Problems der Klassifikation graphischer Information, Vorbereitungen für den späteren Einsatz von Maßbegriffen zu treffen. Generative Graphik hat mit Topologie mehr als mit Numerik zu tun, insofern, als sie ununterbrochen Gestalt- und Lageeigenschaften graphischer Information begegnet oder solche Eigenschaften gezielt erzeugt. Zwischen Topologie und Numerik nun läßt sich ein mathematisches Gebiet einschieben, das zu beiden Bereichen Verbindungen aufnimmt: Die Ordnungstheorie. Um beispielsweise eine bestimmte Klasse von Graphiken nach aufsteigender Mikroinnovation zu ordnen, muß man nicht unbedingt den Meßwert der Mikroinnovation für die einzelne Graphik kennen, ebensowenig, wie man die absoluten Größen von Personen zu wissen braucht, wenn man sie nach steigender Größe geordnet nebeneinander aufstellen will. Nichtsdestoweniger ist ordnende Klassifikation im Sinne mathematischer Ordnung von Wert, ja in vielen Fällen ist die Aufprägung einer Zahlenskala auf die bereits hergestellte Ordnung nur noch eine Frage der Zweckmäßigkeit (wie z. B. bei der Einführung von Notenstufen in der Pädagogik).

Ästhetische Bildinformation überhaupt erschöpfend klassifizieren zu wollen, ist vermutlich ein vergebliches Unterfangen: Jede Klassifikation wird die Entdeckung nichtklassifizierter Fälle nach sich ziehen. Deshalb wird

[1]) Max Bense: Ästhetik und Programmierung, Exakte Ästhetik 5, Stuttgart 1967

eine Betrachtungsweise, die anstückbare und ausbesserungsfähige Klassifikationsnetze über das Areal ästhetischer Informationen wirft, von vornherein in der besseren methodologischen Position sein.

Neben quantitativer oder mindestens ordnender Klassifikation wird in vielen Wissensgebieten auch qualitative benötigt, so auch in der generativen Graphik. Nicht semantisch, sondern schon rein ästhetisch-zeichenhaft ist ja beispielsweise auch ein Baum eine ganz andere Art ästhetischer Information als ein Sandhaufen. Wir verbinden deshalb qualitative mit ordnender Klassifikation. Die Ordnungen, die wir im Bereich der generativen Graphik zu charakterisieren versuchen, nennen wir Gradationen. Was wir im Auge haben, was jedoch in diesem Bericht nur begonnen werden kann, ist ein klassifizierendes Netzwerk für graphische Information, dessen Zweige qualitativ verschiedene Gradationen darstellen. Wir hoffen, damit eine Systematik für Klassifikation gefunden zu haben, die nicht nur für die generative Graphik brauchbar ist, sondern für die Informationsästhetik überhaupt.

Das Klassifikationsproblem der generativen Ästhetik wird durch die Fülle generierbarer Information, vor allem jedoch durch die Reichweite verschiedener Programmierweisen und damit durch die ganze Mannigfaltigkeit möglicher ästhetischer Vorentscheidungen erheblich erschwert. Auch hier haben wir nach einem Ausweg gesucht und ihn — wie wir glauben — zum mindesten im Ansatz gefunden. Der weitaus größte Teil der Bilder, die als Material zur Behandlung des Klassifikationsproblems dienen, wurden mit Hilfe einiger weniger Programme bzw. Transduktoren hervorgebracht. Freilich kann hier der Einwand erhoben werden, daß wir uns

dadurch das Klassifikationsproblem vielleicht unzulässig leichter gemacht haben. Wir können nur mit der Bitte antworten, das fürchterliche Problem der Klassifikation ästhetischer Information auch über diese Untersuchungen hinaus fleißig weiter pflegen zu wollen.

Abschnitt 3.2.1 führt eine Funktion SERIE (QUER, HOCH,XMAL,YMAL,FIGUR) ein, die von fünf Parametern abhängt. In der Programmiertheorie bezeichnet man eine derartige Funktion als Prozedur. Jede Prozedur vertritt ein spezielles, parameterabhängiges Programm. Ersetzt man nämlich z. B. in der Prozedur SERIE die Parameter QUER bis YMAL durch Zahlen a, b, c, d und den Parameter FIGUR wiederum durch eine Prozedur f, so aktiviert der Ausdruck SERIE(a,b,c,d,f) ein ganz bestimmtes Programm, das unter Gültigkeit der Werte a bis f eine ganz bestimmte Bildinformation erzeugt. Ändert man den Wert eines der Parameter ab, so ändert sich das Programm nicht, das *durch den Aufruf* der Prozedur SERIE aktiviert wird, wohl aber kann sich die erzeugte Bildinformation ändern. Der Funktions- oder Prozedur-Ausdruck SERIE(QUER,...,FIGUR) ist gleichzeitig der symbolische Ausdruck für einen Transduktor (auch „black box" genannt), dessen Eingangsinformation Werte der Parameter QUER bis FIGUR und dessen Ausgangsinformation eine Graphik sind.

Um die Wirkungsweise der Prozedur SERIE verstehen zu können, müssen wir einige Bildbeispiele betrachten. Generell ist zu den Bildern in diesem Bericht zu sagen, daß sie alle in kartesische Koordinatensysteme eingezeichnet zu denken sind. Man hat beim einzelnen Bild die Normallage des kartesischen Achsenkreuzes vor sich (X-Achse horizontal, positive Halbachse rechts; Y-Achse vertikal, positive Halbachse oben), wenn man das Bild im

Querformat betrachtet, wobei sich dann die Bildunterschrift gekippt am linken vertikalen Rand befindet. Manche Bilder zeigen nun eine deutliche Gittereinteilung. So kann über die Bilder 22 und 23 ein Gitter von 8mal 6 gleich 48 *Elementarrechtecken* gelegt werden. Bei den Bildern 24 und 25 sind es 3mal 24 gleich 72 und bei Bild 26 sind es 1mal 2 gleich 2 Elementarrechtecke. Das bildeinteilende Gitter, das die Bilder 22 bis 26, jedoch z. B. auch die Bilder 47 bis 51 erkennen lassen, kann nun auf sehr anschauliche Weise mit den ersten vier Parametern der Prozedur SERIE in Verbindung gebracht werden; tatsächlich sind die erwähnten Bilder mit Hilfe von SERIE generiert worden. Die Elementarrechtecke, in die SERIE die Bildfläche gitterartig einteilt (wobei ein Außenbereich des Gitters verbleibt), haben alle die Abmessungen a mal b, wobei a der Wert des Parameters QUER, b der Wert von HOCH ist. Die Anzahl der Elementarrechtecke in Querrichtung (X-Richtung) stimmt mit dem Wert c von XMAL, die Anzahl der Elementarrechtecke in vertikaler Richtung mit dem Wert d von YMAL überein. Man kann also sagen, daß die Prozedur SERIE die Zeichenfläche in c mal d Elementarrechtecke virtuell einteilt, wobei jedes Elementarrechteck die Abmessungen a Längeneinheiten mal b Längeneinheiten hat (beim Originalbild beträgt die Längeneinheit 1 Millimeter). Man beachte nun auch, daß die Namen der Parameter mnemotechnisch gewählt wurden.

Da die Gittereinteilung der Zeichenebene durch SERIE, wie schon gesagt, zunächst nur rein virtuell ist, muß ein zweites prozeßhaftes Element aktiv werden, damit nach Ablauf der Prozedur SERIE überhaupt etwas auf dem Zeichenpapier zu sehen ist. Dazu dient nun der Parameter FIGUR. SERIE zeichnet nämlich in das einzelne Elementarrechteck, oder auch über dessen Grenzen hin-

ausragend, eine Figur ein, die dem obenerwähnten Wert f des Parameters FIGUR entspricht. Wie die Bilder 22 bis 26, jedoch auch die Bilder 47 und 48 deutlich zeigen, sind dabei Zufallsgeneratoren am Werk und man bemerkt rein intuitiv, daß die Mikroinnovation z. B. von Bild 23 größer ist als die von Bild 48. Übrigens ist auch Bild 30 ein Produkt der Prozedur SERIE und an den Bildern 30 und 48 wird der Unterschied der Werte von Mikroinnovation und Redundanz besonders deutlich (man beachte den hohen Redundanzgrad, der ja begrifflich gleichzeitig ein Symmetriegrad ist, in Bild 48).

Abschnitt 3.2.2 ist ganz dem Aufbau einer bestimmten Prozedur ELIRR gewidmet, die als Wert f den Parameter FIGUR in SERIE zu ersetzen vermag. ELIRR zeichnet in das Rahmenrechteck ein geschlossenes Irrwegpolygon (einen ELementaren IRRweg, wie die Abkürzung ELIRR mnemotechnisch andeutet). Den elementaren Irrweg baut ELIRR aus achsenparallelen oder schrägen Strecken auf (Achsenparallelität zeigen die Irrwege in den Bildern 22, 24 und 47, Schräge zeigen die Bilder 23, 25, 26 und 48). Wie wir später sehen werden, generiert SERIE in Kombination mit ELIRR außer den eben erwähnten noch viele andere Bilder, so daß das Bildmaterial zum Hauptabschnitt 3.2 weithin aus einer einzigen genetischen Wurzel stammt. Auf diese Weise wird der Faktor der ästhetischen Vorentscheidung, der ja Heterogenität innerhalb einer Theorie graphischer Information bedeutet, auf ein Minimum herabgedrückt.

Nach diesen Vorbereitungen ist es leicht, die durch SERIE erzeugten Bilder 22 bis 26 als Variationenreihen je eines Themas aufzufassen: Das Thema wird durch den Mechanismus der Prozedur SERIE als Redundanz dispergiert in die Einzelfiguren des Variationenbildes —

soweit Abschnitt 3.2.3. Asymmetrisiert man die Einzelfiguren des Variationenbildes, indem man sie z. B. auffällig längt, so steigert man die Makroinnovation des Gesamtbildes, da sich ein durchgehender Bildnexus bemerkbar macht (beachte die Bilder 24 und 25). Wir bezeichnen diesen Vorgang als Anisotropisierung des Gesamtbildes, dessen Diskretisation durch ihn in eine Gruppierung verwandelt wird (Abschnitt 3.2.4). Man hat hier den Ansatz für eine erste Gradation vor sich, die sich durch Steigerung der Makroinnovation auszeichnet. Bild 26, das den Charakter einer Gestalt oder Doppelgestalt hat, ist ein Exemplar aus der Gradation mit besonders hoher Makroinnovation. Schwächt man dagegen die Makroinnovation eines Variationenbildes, indem man die Gittersymmetrie stört, so erhält man *Haufenbilder* (siehe Bild 27). Daraus folgt die Existenz einer durchgehenden, klassifizierenden Gradation, die mit steigender Makroinnovation von Haufen über Variationen zu Gestalten führt.

In ein qualitativ ganz anderes Teilgebiet führt uns Abschnitt 3.2.5. Ästhetische Information, wie sie uns täglich begegnet, zeigt Konturen. Das Profil eines Gesichts, das Gitterwerk einer Brücke, Umrisse überhaupt, Trennlinien überhaupt: überall handelt es sich hier um Konturen. Was aber innerhalb einer Kontur keine Randkontur mehr zeigt, sondern als ein Zeichenkontinuum an die Konturufer *anbrandet,* nennen wir Textur. Die Texturtheorie führt uns tief hinein in die Mikroästhetik. Abschnitt 3.2.5 kennzeichnet die Texturen als Informationskontinua mit Lokalnexus und ohne Distalnexus. Da wir auch Texturtheorie innerhalb der generativen Graphik betreiben, muß die generative Graphik fähig sein, *Modelltexturen* zu erzeugen. Sie erreicht dies vor allem durch ein Verfahren: Indem sie unter Heranziehung der

Prozedur SERIE dichte Überlappungsfelder von Elementarfiguren erzeugt. Dabei wird die Individualität der Elementarfiguren ausgelöscht und es bleibt der *Lokalnexus* als durchgängiges, qualitativ individuelles Merkmal bestehen. Die Texturen der Bilder 29 bis 36 und 39 bis 42 wurden so gewonnen. Schon am Schluß des einführenden Abschnitts über ästhetische Kategorien sagten wir, daß Zeichenüberlagerungen in der Fläche extrem komplizierte Zeichenverhältnisse schaffen können. Dieser Fall ist bei Texturen die Regel und führt zur Einführung des Begriffs der lokalen Innovationsdichte, um der Strukturkomplexität des Lokalnexus wenigstens durch eine Art von mittelnder Dichtemessung gerecht zu werden. Ein Zusammenhang zwischen der Struktur des Generators einer Textur und ihrer lokalen Innovationsdichte besteht insofern, als die lokale Innovationsdichte um so größer ist, je mehr der Generator Überlappungseffekte begünstigt (siehe Abschnitt 3.2.6). Ist nun die lokale Innovationsdichte eine quantitative oder mindestens ordnungstheoretische Kennzeichnung des Lokalnexus, so erweist sich die typische Qualität des Lokalnexus, die ja eine Invariante quer durch die ganze Textur ist, als die schon weiter oben angekündigte endogene Redundanz.

Die lokale Innovationsdichte, die zusammen mit der Mikroinnovation der Textur wächst oder abnimmt, reiht die Texturen in eine Gradation ein. Die Gradation der Texturen beginnt bei diskretisierten (BENSE würde sagen: *verdampften*) Haufenbildern und endet im unendlich dichten Chaos.

Spezielle Texturtypen, nämlich *Gewirre* und *Gerölle,* werden in Abschnitt 3.2.7 untersucht. Auch das Phänomen der Texturaddition kommt hier zur Sprache. Damit

verläßt die Untersuchung das Gebiet der eigentlichen Texturen. Abschnitt 3.2.8 ist der Erscheinung der Anisotropie in Texturen gewidmet. Anisotropie bringt in Texturen eine durchgehende Ausrichtung von Elementarzeichen und damit Distalnexus mit sich. Distalnexus, als eine globale Bildqualität, bedeutet Makroinnovation. Anisotrope Texturen ordnen sich durch den ihnen innewohnenden Distalnexus in eine Gradation wachsender Makroinnovation ein, die bei distalnexusfreien Texturen beginnt und bei extrem ausgerichteten Bildstrukturen endet. Einen Anfangs-, Zwischen- und Endpunkt dieser Gradation liefern die Bilder 34, 42 und 43. Bild 43 (das auf den Kybernetiker McKay zurückgeht) ist das Modell eines Benseschen *perfekten Kunstwerks.* Vereinigt man Texturen und anisotrope Texturen zu einer einzigen Informationsklasse, so erhält man eine Gradation, die sich zwischen dem Chaos und dem *perfekten Nichtssagendem* aufspannt.

Vom Standpunkt der generativen Ästhetik aus gesehen außerordentlich bemerkenswerte Bildinformationen sind die Gewebe (Abschnitt 3.2.9). Sie zwingen uns, um sie modellmäßig darstellen zu können, sogar zu einer Erweiterung der Befehlssprache, die wir in Kapitel 2 zum Zweck der Genese von Graphiken bereitgestellt hatten. Generiert man nämlich Gewebe (man betrachte die Bilder 44 bis 46), so muß man die Kreuzungspunkte der Fäden (die bei dem von uns verwendeten Gewebebegriff als feste Verschweißungspunkte verstanden werden) schon bei der Genese der Kettenfäden festlegen und zwischenspeichern. Deutet man die Genese von Geweben kosmogonisch, so folgt, daß der Kanal, der gewebeartige Information in einen ästhetischen Kosmos einschleust, komplizierter Zwischenspeicherungen fähig sein muß.

Abschnitt 3.2.10 bespricht die Flächenornamente, deren Entstehung durch einen plötzlichen Zusammenbruch von Mikroinnovation in Texturen gedeutet werden kann; zum anschaulichen Beweis dieser Behauptung vergleiche man die Bildpaare 29—47 und 30—48. Im Fall der Ornamente hat das Grundproblem der generativen Graphik: „Wie bedingt die Struktur des Bildgenerators die Struktur seines Produkts?" eine besonders einfache Lösung: Die Periodizitäten im Ornament sind die sichtbare Spur von periodisch initiierten Zufallsgeneratoren, die — da sie zahlentheoretisch funktionierende Pseudozufallsgeneratoren sind — nach jeder gleichverlaufenden Initiierung auch deterministisch die gleiche Folge von Zufallsgrößen liefern.

Eine Zusammenfassung der im Hauptabschnitt 3.2 erarbeiteten klassifikatorischen Ergebnisse ist das Diagramm in Abschnitt 3.2.11. Es ist der erste Ansatz eines qualitativ-quantitativ klassifizierenden Netzwerkes für die gesamte Computergraphik.

Einiges zum Abschnitt „Gestalt und Gegenstand"

Das Ziel aller genetischen Evolution graphischer Information ist natürlich die Gestalt. Die Aesthetica sagen zum Gestaltproblem (Seite 247): „Das Zeichen als differenziertes Gebilde, als Element kann einem integrierenden ästhetischen Prozeß unterliegen, dann entsteht Ganzheit, Gestalt." Wesentlich zur Integration und Stabilisierung eines Zeichensystems trägt jedoch der Zusammenhang seiner Komponenten, auch über Entfernungen hinweg, bei. Aus diesem Grunde rechtfertigt sich jetzt die vorläufige Definition des Gestaltbegriffs, wie sie im Abschnitt 3.2.5 über die Texturen gegeben wurde: „Gestalten sind ästhetische Informationseinheiten mit Lokal-

und Distalnexus." Beispiele von Gestalten finden sich zahlreich in den Bildern zu diesem Bericht. Schon die Einzelfiguren der Variationenbilder (Bilder 22 bis 26) konstituieren ja Gestalten; Einzelgestalten sind in den Bildern 6 bis 9, 13, 14, 17, 19, 20, 42, 51 und 52 zu sehen. Bild 51 zeigt, daß Distalnexus von Zeichenkomponenten nicht notwendig topologischen Zusammenhang einschließen muß.

1.4. Weitere Probleme

Die vorliegende Arbeit beschränkt sich auf qualitative oder ordnungstheoretische Analysen der synthetisierten Graphiken. Wir sagen zwar manchmal, daß die Mikroinnovation einer bestimmten Graphik größer sei als die einer anderen, machen jedoch keine expliziten Zahlenangaben über die Größe der Mikroinnovation. Die Einführung von quantitativen Maßbegriffen in die generative Graphik macht Schwierigkeiten. Am ehesten ist noch die *lokale Innovationsdichte,* die wir in Abschnitt 3.2.6 definieren werden, geeignet zu einer Quantifikation. In jenem Abschnitt wird nämlich gesagt, daß die lokale Innovationsdichte die Lagerungsdichte der Protozeichen in einer Textur sei. Dabei sind die Protozeichen Spezialfälle der von Bense angeführten *syntaktischen Partikel* (siehe AE, Seite 143). Ist also die lokale Innovationsdichte die Anzahl der Protozeichen pro Flächeneinheit, so ist zunächst einmal die Frage, was im Fall einer bestimmten Textur überhaupt als Protozeichen zu gelten hat. Soll man z. B. bei Bild 37 auch die schwachen Knicke in der Linienführung, die dem Betrachter Kreisbögen vortäuschen, mit zu den Protozeichen rechnen, oder sind allein die Kreisbogenabschnitte authentische Protozeichen?

Das Problem der quantitativen Bemessung von maschi-

nell generierten Graphiken gestaltet sich noch wesentlich komplizierter, wenn man es unter Beachtung der Bedingung anzupacken versucht, daß die Eigenschaften der Graphik aus den Eigenschaften des generierenden Algorithmus erschließbar sein müssen. Dann müßte man nämlich ein Programm schreiben, das Algorithmen als textartige Eingangsinformation akzeptiert und Zahlenangaben über die Eigenschaften der von den analysierten Algorithmen generierbaren Graphiken als Ausgangsinformation abgibt. Nun ist bemerkenswert, daß z. B. die beiden Teilbilder von Bild 17 (das wir schon in Abschnitt 1.3 „Übersicht über das Kapitel 3" erwähnt haben) durch den gleichen Algorithmus generiert werden können. Je nachdem, wie man in dem erwähnten Algorithmus die Schwankungsbreite der Kreise wählt, die er zu Papier bringt, erhält man qualitativ ganz verschiedene Graphiken. Das Kreisüberschneidungssystem des unteren Teilbildes von Bild 17 ist nicht aus der Struktur des generierenden Algorithmus vorhersagbar. Schuld daran sind unter anderem die Zufallsgeneratoren, die zur Genese der beiden Teilbilder herangezogen werden und Teil des generierenden Algorithmus sind.

Eine quantitative Analyse von maschinell erzeugten Graphiken könnte sich also frühestens an den fertigen Graphiken entfalten. Damit wäre jedoch der engere Bereich der generativen Graphik und Ästhetik bereits überschritten. Verf. plädiert dafür, daß zukünftige Untersuchungen die generative Graphik von Anfang an in eine volle Kommunikationsästhetik einbetten.

Wie schon in der Übersicht über das dritte Kapitel gesagt wurde, müßte eine solche Forschungsrichtung drei miteinander gekoppelte Informationsmaschinen untersuchen: Informationsgenerator, Informationstransporteur

(Informationskanal), Informationsanalysator. In diesem Rahmen könnte zweifellos auch das ästhetische Maßproblem für die generative Graphik leichter gelöst werden. Um anschaulich zu machen, welcher kostspieligen Mittel man sich zu bedienen hätte, sei etwas weiter ausgeholt: Olaf Stapledon berichtet in seinem Roman „Odd John" von dem Mädchen Jelly, das einen feinkörnigeren Formsinn als andere Menschen besaß. Für Jelly war das zunächst sehr unangenehm, da ihr kein Artefakt in der Gestalt erschien, die der Hersteller beabsichtigt hatte. Kunst war für sie eine Marter, wegen der Ungenauigkeit sowohl der Auffassung als auch der Ausführung der Kunstobjekte. Der Held des Romans führt das alles auf eine feinkörnigere Retina in Jellys Augen zurück (Olaf Stapledon: Odd John, New York 1959; Stapledon muß übrigens als einer der großen Utopisten des 20. Jahrhunderts angesehen werden, W. Grey Walter, der Erfinder der künstlichen Schildkröte, nannte ihn den Virgil der Science Fiction).

Wir machen nun den Vorschlag, die Fähigkeit des Computers zur raschen Analyse komplexer Vorgänge, zur Simulation von Sehapparaten im Hinblick auf die Probleme der Informationsästhetik heranzuziehen. Von den drei Maschinen, die im Rahmen einer vollen Kommunikationsästhetik zu studieren sind, wäre dann auch die letzte, nämlich der Informationsanalysator auf dem Weg zur praktischen Realisierung und experimentellen Nutzung — vorliegender Bericht zeigt ja die Realisierbarkeit des Informationsgenerators, während als Informationskanal der äußere optische Kanal (evtl. durch Linsensysteme ergänzt) zu benutzen wäre. Als unmittelbar erhältliche technische Einrichtung könnte der sogenannte Light-Pen dienen: er besteht aus einem Griffel, mit dem man auf die Oberfläche eines Bildschirms schreiben

kann. Der Bildschirm ist an einen leistungsfähigen Computer angeschlossen. Alle zeichenartige Information, die mit dem Light-Pen auf den Bildschirm geschrieben wird, erscheint sofort sichtbar auf dessen Oberfläche und wird gleichzeitig im Gedächtnis des Computers abgespeichert. Der Computer kann derartig programmiert werden, daß er seinerseits Antwortinformation auf dem Bildschirm aufzeichnet. Mit dieser Anordnung kann die Retina mit ihren nachgeschalteten Zentren, wenn auch nicht vollständig, so doch im Hinblick auf gewisse Informationsprozesse, die den Ästhetiker interessieren, nachgebildet werden. Man könnte z. B. der folgenden Frage nachgehen: Was wird aus ästhetischer Information, wenn ein Objekt in das interne Symbolsystem eines Sehrezeptakulums übersetzt wird? Bleibt dabei überhaupt optisch-anschauliche Information erhalten und kann man sinnvoll von ihr sprechen? Wie sieht ein Simulationsprogramm aus, das dem Kreisüberschneidungssystem aus Bild 17 gewachsen ist und es in seine Komponenten zerlegen kann?

Folgendes Problem schließt sich den anderen an: Wenn die menschliche Retina feinkörniger ist, als sie es zum Zweck der Bewältigung lebensnotwendiger Vorgänge sein müßte, dient dann ihr *Überschuß* grundsätzlich zur Bewältigung ästhetischer Information? Diese physiologische Frage läßt sich rein ästhetisch-kosmogonisch vielleicht so fassen: Setzt die Evolution ästhetischer Information im Rahmen eines philosophisch-kosmogonischen Modells ein ständiges Plus oder *Voraus* an informationsanalytischer Potenz voraus? Eine weitere Frage ist die, ob die Feinkörnigkeit der Retina im tierischen Auge teilweise brachliegt und ob es der typischen humanen Zentren des Gehirns bedarf, um den Überschuß zur Verarbeitung ästhetischer Information heranzuziehen.

Übrigens ist auf dem Umschlag der schon von uns zitierten Literaturstelle „Exakte Ästhetik 5" ein Light-Pen-Gerät zu sehen. Freilich ist der Anschaffungswert derartiger Ausrüstung des ästhetischen Laboratoriums ungewohnt hoch. Wenn man jedoch bedenkt, welch große Bedeutung die Analyse ästhetischer Information auch für Design-Zwecke, für Arbeitspsychologie und Pädagogik hat, dann ist die Installierung von „Large-Scale-Forschungsgerät" vom unmittelbar lebenspraktischen Standpunkt aus gerechtfertigt.

2. Eine Programmiersprache für die Genese von Graphiken

Der moderne Sprachbegriff unterscheidet natürliche Sprachen und Kunstsprachen. Eine große Anzahl und einen besonderen Typus von Kunstsprachen hat die Logistik hervorgebracht. Jeweils eine Sprache der Logistik spricht über einen ganz bestimmten Individuenbereich. Die Sprache der Arithmetik beispielsweise, wie sie in der Logistik eingeführt wird, spricht über die natürlichen Zahlen. Basis jeder logistischen Sprache sind gewisse Grundbegriffe. Die Grundbegriffe der Arithmetik sind die natürliche Zahl, die Eins und der Nachfolger einer natürlichen Zahl. Man verwendet die Grundbegriffe einer logistischen Sprache zur Formulierung von Grundsätzen, die auch als die Axiome der Sprache bezeichnet werden. Mit Hilfe von Ableitungsregeln gewinnt man aus den Axiomen Theoreme. Die Klasse der Theoreme einer logistischen Sprache ist im wesentlichen die Klasse der wahren Aussagen über die Individuen, von denen die Sprache spricht.

Da wir auf dem Standpunkt der Informationstheorie stehen, interessiert uns die Beziehung zwischen Informationstheorie und Logistik. Sie besteht darin, daß es der Logistik gelingt, mindestens einen außerordentlich großen Umfang an Wissen über einen bestimmten Individuenbereich durch die Verwendung der Ableitungsregeln auf eine geringe Anzahl von Grundbegriffen und Axiomen zurückzuführen. Im Fall der Arithmetik sind die Axiome die Aussagen, daß die Eins eine natürliche Zahl und nicht Nachfolger einer natürlichen Zahl ist, daß

jede natürliche Zahl einen eindeutig bestimmten Nachfolger hat, verschiedene natürliche Zahlen jedoch verschiedene natürliche Zahlen als Nachfolger besitzen und schließlich, daß eine Eigenschaft, die der Eins zukommt und mit jeder natürlichen Zahl auch ihrem Nachfolger, jeder natürlichen Zahl zukommt. Man hat hier nichts anderes vor sich als Informationsreduktion, Beseitigung jeder Redundanz, Codierung jeder wesentlichen Wahrheit der logistischen Sprache durch eine Kombination der Grundbegriffe und Axiome mit einer bestimmten Ableitung.

In neuerer Zeit ist neben die Betrachtung von Individuenbereichen die Betrachtung einer anderen Art von Entitäten getreten, nämlich die Untersuchung von zeitlichen Prozessen. Individuen und Beziehungen zwischen Individuen werden durch Theoreme beschrieben, Prozesse durch Algorithmen. In der Mathematik sind Algorithmen seit langem bekannt, so z. B. der Euklidische Algorithmus, der den Prozeß zur Gewinnung des größten gemeinsamen Teilers zweier natürlicher Zahlen beschreibt. Die systematische Erforschung der Beziehungen zwischen Prozessen und Algorithmen ist jedoch durch einen Neuankömmling unter den Erkenntniswerkzeugen des Menschen außerordentlich gefördert worden: Wir meinen den Computer, auch Datenverarbeitungsanlage — abgekürzt DVA — genannt. Die DVA erlaubt das Durchspielen mathematischer Prozesse mit sehr hoher Geschwindigkeit bei gleichzeitiger Aktivierung eines sehr umfangreichen, mit dem Prozeß verknüpften Datenmaterials. Durch die Erwähnung des Datenbegriffs werden wir zur Informationstheorie zurückgeführt. Tatsächlich verarbeitet die DVA Information, sie ist ein Informationswandler. In die DVA strömt Information hinein und heraus. Den Algorithmus, der den Pro-

zeß des Informationsumsatzes in der DVA beschreibt, bezeichnet man als das Programm der DVA. Jede DVA kann auf eine große Anzahl von Programmen umgestellt werden. Man kann sogar sagen, daß das Programm der DVA zu den Daten gehört, die sie verarbeitet. Das Programm, das die DVA zu ihrem augenblicklichen Verhalten befähigt, wird nämlich vor Anlaufen des Prozesses in den Datenspeicher der DVA eingeschrieben und steuert nun von dort aus den Prozeß, wobei der tatsächliche Ablauf jedoch noch vom Zusammenwirken des Programms mit der Eingangsinformation der DVA abhängt.

Es ist ein leichtes, die Ausgangsinformation der DVA auf technische Einrichtungen wirken zu lassen, die bestimmte Aktionen vollführen. In Anlehnung an die Biologie kann man nun die Elemente der DVA, die Eingangsinformation aufnehmen, als die Rezeptoren der DVA, die Elemente jedoch, die mit Hilfe der Ausgangsinformation agieren, als die Effektoren der DVA bezeichnen. Auf diese Weise gelangt man vom Begriff der DVA zum Begriff des Universalautomaten. Durch das jeweilige Programm im Datenspeicher der DVA wird sie — der Universalautomat — auf ein jeweilig besonderes rezipierendes und effizierendes Verhalten umgestellt. Ist der Effektor eine Einrichtung zum Anfertigen von Zeichnungen, während der Rezeptor gewisse Daten über diese Zeichnungen aufnimmt, so haben wir den universalen Zeichenautomaten vor uns, wie er uns später sehr viel beschäftigen wird.

Von jetzt ab also begreifen wir die DVA als einen Universalautomaten, umstellbar auf verschiedene Aufgaben durch verschiedene Programme. Auch die Programme nimmt der Universalautomat durch seine Rezeptoren auf. Bei fast allen technisch realisierten Universalauto-

maten ist der hauptsächliche programm- und datenaufnehmende Rezeptor eine Lochkartenabfühlstation. In die nacheinander am Rezeptor vorbeiwandernden Lochkarten ist die Information der Daten und Programme in Form von besonders verteilten Löchern eingestanzt.

Solange ein ganz bestimmtes Programm den Universalautomaten beherrscht und solange man von seinen Fähigkeiten absieht, auch noch ganz andere Programme zu aktualisieren, ist der Universalautomat eingeschränkt zum speziellen Automaten. Wir wollen uns daran gewöhnen, jedes Programm, unter dem Aspekt seiner Aktualisierbarkeit, d. h. seiner zeitlichen Verwirklichbarkeit, als äquivalent anzusehen mit einem in Tätigkeit befindlichen speziellen Automaten, der im Ablauf seiner zentralen und peripheren Prozesse gerade dieses spezielle Programm aktualisiert. Umgekehrt nehmen wir von jedem Automaten an, daß ein Programm existiert, das er aktualisiert. Selbstverständlich machen wir als Informationstheoretiker keinen Unterschied zwischen den von Menschen gebauten Automaten und den Lebewesen, zu denen auch der Mensch gehört. Auch dem Menschen entspricht ein Programm, das er von der Geburt bis zum Tod realisiert.

Dieses Äquivalenzprinzip zwischen Automaten und den ihnen zugeordneten Programmen hat bedeutende Konsequenzen für die Anforderungen an die von Menschen zum Zweck der Steuerung von Automaten geschriebenen und zum Beispiel in Lochkarten gestanzten Programme. Nach dem Äquivalenzprinzip ist jedes solche Programm äquivalent mit einer menschlichen Person, die das Programm versteht, indem sie es mindestens in Gedanken Schritt für Schritt nachvollzieht, ferner ist das Programm äquivalent mit jedem es aktualisierenden

technisch realisierten Automaten. Also sind auch interpretierender Mensch und realisierender technischer Automat miteinander äquivalent. Natürlich kann das Äquivalenzprinzip nur gelten, wenn Mensch und technischer Automat beide ideal gut funktionieren. Will man, daß es gilt, so verwandelt sich das Äquivalenzprinzip in eine Norm und aus dieser Norm folgt eine zweite, die wir als das Exaktheitsprinzip bezeichnen: Das Programm muß so ausführlich und eindeutig abgefaßt sein, daß es vom interpretierenden Menschen und aktualisierenden Automaten in der gleichen Weise verstanden wird.

Das Exaktheitsprinzip führt uns zurück zum Begriff der Kunstsprache. Denn die geforderte Exaktheit kann mit Sätzen der Umgangssprache nicht erreicht werden. So wie die Kunstsprachen der Logistik die exakte Ableitung der logischen Konsequenzen aus Axiomensystemen ermöglichen, erlauben die Kunstsprachen der Programmiertheorie, man nennt diese Kunstsprachen Programmiersprachen, die Formulierung von Programmen in so exakter Weise, daß sie zum übereinstimmenden Verhalten aller sie interpretierenden Automaten führen, seien diese Automaten Menschen oder Maschinen.

In indikativischer Weise sprechen die Sätze der logistischen Sprachen über Sachverhalte. Die in Programmiersprachen formulierten Programme fordern vom Automaten ein bestimmtes Verhalten, sie sind deshalb nicht indikativisch, sondern imperativisch aufgebaut. Programme sind also Handlungsanweisungen für Automaten. Die Absätze, aus denen Programme zusammengesetzt sind, bezeichnet man schlicht als Anweisungen. Anweisungen schließen sich zusammen zu Blöcken und Prozeduren und die Prozeduren stufen sich schließlich zu fertigen Programmen auf. Dieser hierarchische Bau

der Programme hat seine genaue Entsprechung im hierarchischen Bau des Verhaltens von Automaten. Betrachten wir z. B. die Entstehung eines Landschaftsbildes. Hier zerfällt das Verhalten in Abschnitte, die zur Realisierung von Bäumen oder Häusern führen, in Häuser werden Fenster eingezeichnet und als unterste Verhaltensstufe finden wir die Ausführung der einzelnen Zeichenstriche, wobei diese Grenze durchaus relativ ist, denn man könnte nach unten zu Muskelkontraktionen und Innervationsimpulsen fortschreiten.

Unter den vielen Programmiersprachen, die seit dem Ende des zweiten Weltkrieges entstanden sind, hat sich eine als ganz besonders geeignet für Grundlagenexperimente herausgestellt. Diese Sprache heißt ALGOL, der Name „ALGOL" ist ein aus *Algorithmic Language* gebildetes Acronym. Wir werden eine Erweiterungssprache der als ALGOL 60 bezeichneten Fassung von ALGOL entwickeln und benutzen[1]). Die Betrachtungen in den nachfolgenden Abschnitten schließen sogar einen ALGOL-Kurs ein, der jedenfalls so weit führt, als es für unsere Zwecke notwendig ist.

Jene Erweiterungssprache zu ALGOL, die es ermöglichen soll, Programme für die automatische Genese von Graphiken zu schreiben, nennen wir G. Die Sprache G baut sich aus einzelnen Stufen verschiedener Kompliziertheit und dementsprechend verschiedenen Ausdrucksvermögen auf. Die erste und basalste Stufe G1 beschreiben wir im nächsten Abschnitt. G1 wird uns befähigen, alle Graphiken darzustellen, die schrittweise aus geradlinigen Strecken verschiedener Länge erzeugt werden können. Das sind sogar alle Graphiken über-

[1]) Revised Report on the Algorithmic Language ALGOL 60, Springer-Verlag, Berlin 1963

haupt, wenn man auch Strecken sehr kleiner Länge in Betracht zieht. Mit Hilfe des Äquivalenzprinzips anders gewandt kann man auch sagen, daß der nächste Abschnitt eine Klasse von speziellen Zeichenautomaten beschreibt, von denen jeder eine Graphik generiert, die einen zusammenhängenden oder in Komponenten zerfallenden geradlinigen Graphen — einen Streckenkomplex — darstellt.

Wiederholen wir: Programme und spezielle Automaten sind die statischen und dynamischen Aspekte von Prozessen. Begrifflich ist das Programm äquivalent mit allen Automaten, die es realisieren (Äquivalenzprinzip). Ein Programm als die Darstellung eines Prozesses in einer Programmiersprache muß so evident dargestellt sein, daß es von allen Automaten aus der Klasse der mit dem Programm äquivalenten Automaten in gleicher Weise interpretiert wird (Exaktheitsprinzip).

2.1. Die Genese von Streckenkomplexen

Unter den Automaten bilden die Zeichenautomaten eine besondere Klasse: Sie sind die Automaten, die ein Zeichengerät als Effektor besitzen. Technisch sind die Zeichenautomaten im allgemeinen als Universalautomaten realisiert, d. h. sie werden durch Eingabe spezieller Programme zur Erzeugung jeweils einer Zeichnung befähigt. Der begrifflichen Analyse fällt es leichter, jedem Programm einen eigenen Zeichenautomaten zuzuordnen, der es aktualisiert. Für uns wird also jede Graphik durch ihren Automaten generiert; dieser ihr Automat wird von einem Zeichenprogramm gesteuert.

Wir haben hier eine Abbildung der erzeugten Bilder der generativen Graphik auf die Zeichenautomaten vor uns.

Diese Abbildung ist allerdings einmehrdeutig, d. h. eine bestimmte Graphik kann immer durch mehrere verschiedene Programme erzeugt werden. Beispielsweise können erst die waagrechten Seiten eines Rechtecks, dann die senkrechten gezeichnet, oder es kann auch umgekehrt verfahren werden.

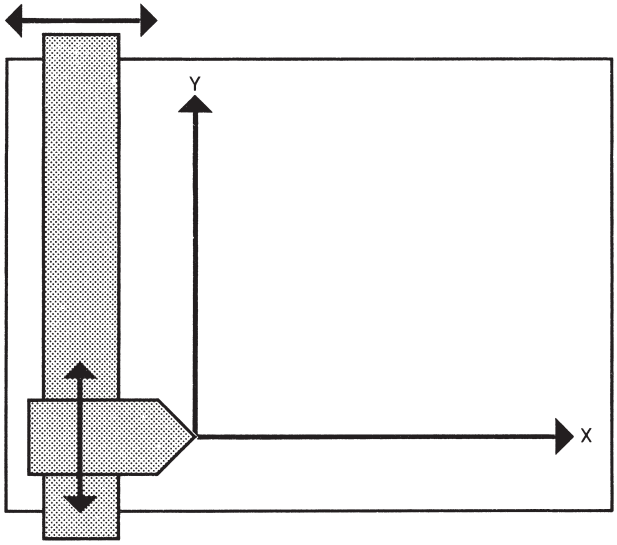
Über den technischen Aufbau von Zeichenautomaten werden wir nicht viel zu sagen haben, er ist verhältnismäßig einfach: Der Zeicheneffektor besteht aus einem rechteckigen Tisch, auf dem das Zeichenpapier befestigt wird. Auf dem Zeichenpapier bewegt sich ein Griffel, der von einem motorisch betätigten Gestänge bewegt wird. Aus einer elektronischen Steuereinheit erhalten die Motoren des Zeichentisches ihre adäquate Bewegungsinformation. Die zwei Kanten des rechteckigen Zeichentisches entsprechen den zwei Achsen eines rechtwinkligen Koordinatensystems; das Gestänge vermag den Griffel an jeden gewünschten Ort in diesem Koordinatensystem zu bringen, der innerhalb des Rahmens des Zeichentisches liegt. Da wir uns den Zeichenautomaten als einen speziellen Automaten mit speziellem inkorporiertem Programm denken, benötigt der Zeichenautomat nur einen einzigen Rezeptor, nämlich einen Startknopf, der den Zeichenprozeß auszulösen gestattet (höchstens innerhalb einer bestimmten Ontologie könnten wir die Welt der Erscheinungen uns vorgezeichnet denken durch einen Zeichenautomaten, der ohne zeitlichen Anfang schon immer zeichnet).

Wie sehen nun die Programme der Sprache G1 aus, mit deren Hilfe die Zeichenautomaten Graphiken generieren? Der Unterabschnitt 2.1.1 wird uns zunächst zeigen, daß das Exaktheitsprinzip einen ganz bestimmten Aufbau dieser Programme erzwingt.

2.1.1. Die Grundanweisungen OPEN, LEER, LINE und CLOSE der Sprache G1

Beim Entwurf eines Zeichenprogramms passen wir uns der technischen Konstruktion des Zeichenautomaten an, indem wir uns die zeichnerische Information immer auf ein rechtwinkliges Koordinatensystem bezogen denken, das zu den Kanten des Zeichentisches parallel liegt. Wir nennen dieses Koordinatensystem das XY-Koordinatensystem. Es ist nun merkwürdig, daß die Einführung des XY-Koordinatensystems zusammen mit dem Exaktheitsprinzip einen Zwang auf die Eröffnungsweise des Zeichenprogramms nach sich zieht. Auf jeden Fall hat ja der Griffel zu Beginn der Zeichnungsgenese eine ganz bestimmte Lage im XY-System. Das Wissen über diese Lage ist eine Information, die der Entwerfer und damit erste Interpretationsautomat des Programms in Besitz hat. Diese Information muß klar und eindeutig im Programm ausgedrückt werden, sonst kann die generative Aktivität des Zeichenautomaten nicht mit der interpretierenden Aktivität des Entwerfers konform gehen. Umgangssprachlich lautet die Information über die Anfangslage des Griffels etwa: *Der Griffel hat die Koordinaten X=A und Y=B im XY-System.* Dabei sind A und B angemessene Koordinatenwerte, also reelle Zahlen. Für den Entwerfer, der eine Skizze der zu generierenden Graphik macht, hat die Anfangsinformation den Sinn: *Der Griffel in meiner Hand hat die Koordinaten X=A und Y=B in dem Koordinatensystem, dem ich Zeichenpapier und Skizze einzuordnen gedenke.* Für den Zeichenautomaten — und wir kommen natürlich nicht umhin, hier anthropomorphe Theorie des Automaten zu treiben — hat die Information den Sinn: *Gestänge und Griffel meines Zeicheneffektors haben die Anfangskoordinaten X=A und Y=B im XY-System.* Meistens befindet sich der Griffel zu Beginn des Programmablaufs in der linken unteren Ecke des Zeichentisches. Dann ist es z. B. angemessen, den Koordinatengrößen A

und B die Werte 0 zuzuordnen. In diesem Fall laufen die Achsen des XY-Systems durch einen festen Punkt in der linken unteren Ecke des Zeichentisches (siehe Figur 1).



Figur 1

Figur 1 zeigt den Tisch des Zeichenautomaten in Draufsicht. Durch Doppelpfeile sind die Bewegungsmöglichkeiten der beiden gegeneinander verschieblichen Teile des Gestänges angedeutet. Der Ort des Griffels und der Ursprung des Koordinatensystems stimmen im Fall A=0, B=0 überein. Wir wollen diese Abstimmung der Anfangslage des Griffels auf die Lage des XY-Systems als Eröffnung bezeichnen und in der Sprache G1 durch die Anweisung

(1) OPEN(A,B)

wiedergeben. Im Fall von Figur 1 beschreibt also G1 die Eröffnung durch die Anweisung

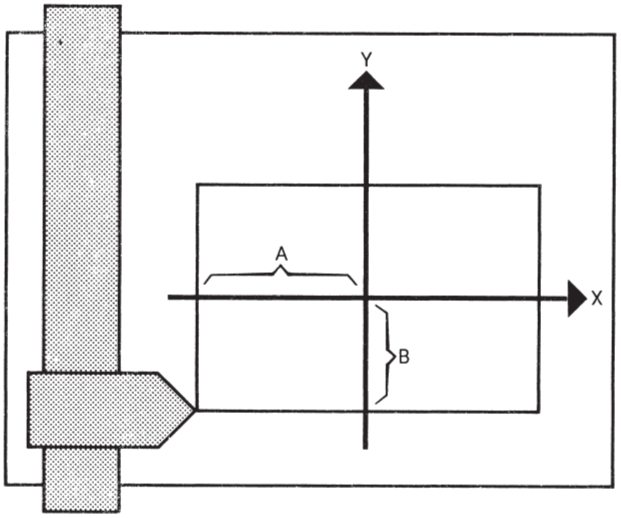
(2) OPEN(0,0)

In der Theorie der Sprache ALGOL werden Programme, die wohlabgegrenzte Ausschnitte aus Prozessen wiedergeben, als Prozeduren bezeichnet. OPEN ist eine solche Prozedur und OPEN(A,B) ist der Aufruf, d. h. die Aktualisierungsanweisung, von OPEN in allgemeiner Form. A und B heißen die Formalparameter der Prozedur. In einem nicht allgemein, sondern faktisch vorzunehmenden Prozeduraufruf werden die Formalparameter durch Werte ersetzt, die Aktualparameter genannt werden. Im Sinn dieser Definition ist der Ausdruck (2) ein Prozeduraufruf der Prozedur OPEN, wobei beide Aktualparameter den Wert 0 annehmen.

In vielen Fällen entspricht es nicht der Absicht des Zeichenprogrammentwerfers, die Anfangslage des Griffels in den Koordinatenursprung zu verlegen. Figur 2 zeigt den besonders interessanten Fall, in dem der Griffel zwar zu Anfang auf die linke untere Ecke eines Zeichenkartons weist, der auf dem Zeichentisch liegt, das Koordinatenkreuz jedoch durch die Mitte des Kartons verlaufen soll. Jetzt dürfen A und B in (1) natürlich nicht die Aktualwerte 0 erhalten. Im Fall eines DIN-A 4-Zeichenblatts ist

(3) OPEN(−148.0,−105.0)

die richtige Eröffnung, wobei als Längeneinheit das Millimeter dient. Der Ausdruck (3) befolgt die Übereinkunft in einem Spezialfall, daß wir reelle Aktualparameter immer mit dem Dezimalpunkt schreiben wollen, auch wenn dieser nur die Null anführt. Eine Ausnahme macht nur die reelle Zahl 0 selbst.



Figur 2

Übrigens können in einem Zeichenprogramm beliebig viele Eröffnungen nacheinander angewiesen werden, von denen jede das XY-System an eine andere Stelle rückt. Bei komplizierten Graphen mit vielen Komponenten kann das die Programmierung beträchtlich vereinfachen.

Die Eröffnung hat ihre Ergänzung in der Schließung, die gewisse Abschaltprozesse im Zeicheneffektor des Automaten bewirkt. Die Schließung wird in der Sprache G1 durch die parameterlose Prozedur CLOSE wiedergegeben. Im Sinn des Exaktheitsprinzips sagt CLOSE auch zum menschlichen Entwerfer wohlverständlich: *Lege den Zeichenstift weg!*

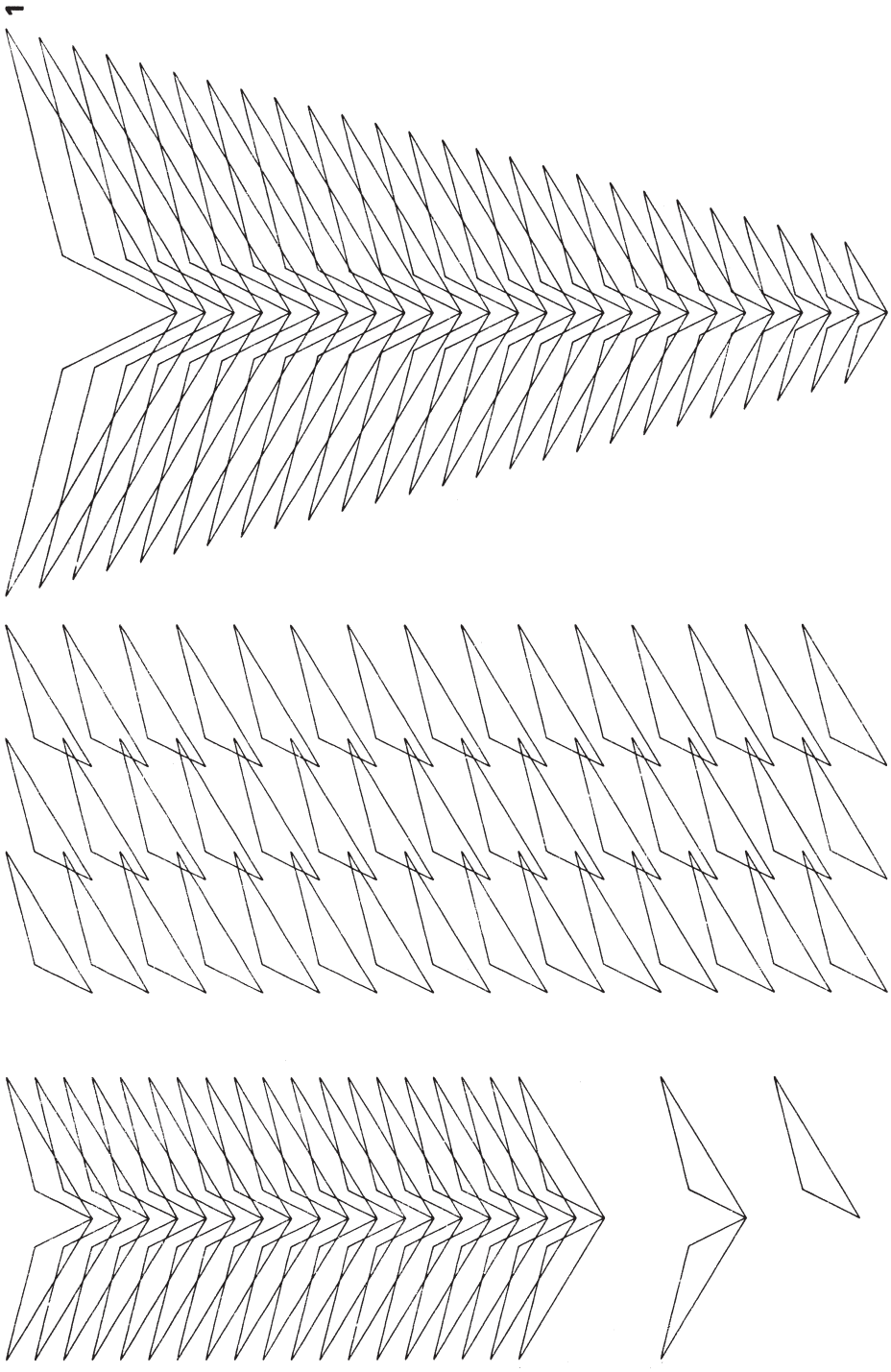
Noch haben wir — als die Entwerferautomaten der Zeichnung — nichts gezeichnet. Wir springen jetzt je-

doch mitten in einen Zeichenprozeß hinein. Bild 1 ist im Querformat zu betrachten. In der linken unteren Ecke von Bild 1 befindet sich der rechte Flügel einer einfachen Falterfigur. Diese Teilzeichnung wird durch das folgende Programm bewirkt:

```
(4)  1 'BEGIN'  OPEN(0,0).,
     2          LEER(40.0,15.0).,
     3          LINE(65.0,30.0).,
     4          LINE(45.0,25.0).,
     5          LINE(40.0,15.0).,
     6          CLOSE
     7 'END'
```

Um auf die einzelnen Zeilen von Programm (4) bequem Bezug nehmen zu können, haben wir es mit einer Zeilenzählung versehen, die nicht selbst zum Programm gehört. Wie jedes Programm, das wir in der Folge studieren werden, beginnt Programm (4) mit der Zeichenfolge 'BEGIN' und endet mit der Zeichenfolge 'END'. Solche Zeichenfolgen, die mit Apostrophen anfangen und aufhören, bezeichnet die ALGOL-Theorie als Wortsymbole. Wortsymbole haben den gleichen grammatikalischen Charakter, wie etwa die Klammern oder der Punkt, d. h. sie sind in der Sprache ALGOL nicht weiter sinnvoll zerlegbar. Eigentlich stellen sie eine Verlegenheitslösung dar: Man könnte, wenn die Typenvorräte der — auch von den Automaten gesteuerten — Schreibmaschinen größer wären, irgendwelche Sonderzeichen anstatt der Wortsymbole wählen. Auf 'BEGIN' folgt im Programm (4) die uns schon wohlbekannte Eröffnung. Zwischen je zwei Anweisungen des Programms steht die aus den Zeichen Punkt und Komma bestehende Zeichenfolge, die wir als Ersatz für den Strichpunkt benutzen. Bei dieser Schreibweise lehnen wir uns an den Typensatz des

Bild 1



Druckeffektors der Datenverarbeitungsanlage an, die Verf. für seine Experimente benutzte. Der Druckeffektor ist hier eine mit hoher Geschwindigkeit zeilenweise druckende elektrische Schreibmaschine, die jedoch weder über den Strichpunkt noch über eine ganze Reihe anderer Sonderzeichen verfügt.

Fahren wir nun in unserer Analyse von Programm (4) fort! Zeile 2 erteilt das erste Kommando an den Zeichentisch, die Zeilen 3 bis 5 erteilen weitere Kommandos. Diese Kommandos sind Anweisungen der Art

(5) LEER(U,V)

und

(6) LINE(U,V)

Die Wirkungsweise von (5) und (6) ist außerordentlich einfach: Beide Anweisungen bewegen den Griffel von seiner derzeitigen Position zu der neuen Position mit den Koordinaten U,V im XY-Koordinatensystem. Dabei besteht jedoch ein sehr wichtiger Unterschied zwischen LEER und LINE. Der Prozeduraufruf LEER(U,V) bewegt nämlich, im Gegensatz zum Aufruf LINE(U,V), den Griffel über dem Zeichenpapier, ohne eine sichtbare Spur zu hinterlassen. Man kann die Griffelbewegung mit Hilfe von LEER, das Leerfahren des Griffels, auch als die Griffelpositionierung im Gegensatz zum eigentlichen Strichziehen bezeichnen. Jetzt ist der Sinn von Programm (4) klar: Zeile 2 positioniert den Griffel, und die Zeilen 3, 4 und 5 ziehen das Dreieck des rechten Falterflügels. Die Schließung CLOSE und das 'END'-Wortsymbol schließen das Programm ab.

Der Leser kann nun nach Belieben den Sinn des Exaktheitsprinzips dadurch erproben, daß er selbst den Zeichenstift nimmt, sich in einen Interpretationsautomaten verwandelt und Programm (4) schrittweise nachvollzieht. Bild 1 ist nur eine Kopie der Originalzeichnung des Zeichenautomaten, kann also eine Maßstabsveränderung enthalten. In der Originalzeichnung beträgt die Flügelspannweite der größten Falterfigur in Bild 1, es ist die im Bild ganz rechts oben, genau 100 Millimeter.

2.1.2.
Relativisierung der Lage von Komponenten einer Graphik

Programm (4/2.1.1) enthält alle Grundanweisungen, durch die das Sprachkonzept G1 über die Programmiersprache ALGOL hinausgeht. Man könnte nun alle Streckenkomplexe geduldig durch Programmzeilen aufbauen, wie sie in Programm (4/2.1.1) verwendet wurden. Betrachtet man jedoch die verhältnismäßig kompliziert gebauten übrigen Teilfiguren in Bild 1, so möchte man schon wegen der manuellen Schreibarbeit verzweifeln, die mit dem Aufbau der zugehörigen Programme verbunden wäre, gäbe es nicht Ausdrucksmittel in ALGOL, die mühelos über den schwerfälligen Programmierstil unseres ersten Programms hinausführen. Diese neuen Ausdrucksmittel nützen ausnahmslos Redundanz in den zu beschreibenden Zeichenprozessen und Streckenkomplexen. Auf welche Weise das vor sich geht, werden wir Schritt für Schritt besprechen. Die mit der Ausnützung von Redundanz verbundene Informationsreduktion macht die Sprache G1 so interessant für die neue Ästhetik und ihre Reflexionen.

Überflüssige Information enthält Programm (4/2.1.1) insofern, als in allen LEER- und LINE-Anweisungen, wir bezeichnen sie auch als die Fahranweisungen des Pro-

gramms, absolute Koordinatenwerte stehen. Man kann nämlich genausogut in den Zeilen 2 bis 5 des Programms einen Teil der Zahlenwerte durch Variablen ersetzen. Je nach dem Zahlenwert, den man den Variablen zuweist, bewegt sich dann der ganze rechte Flügel der Falterfigur von Stelle zu Stelle im Koordinatensystem. Bevor wir diese umgangssprachlichen Aussagen in Anweisungen der Sprache G1 umwandeln können, müssen wir jedoch den Begriff der Wertzuweisung an eine ALGOL-Variable besprechen.

Nehmen wir an, daß wir den Fußpunkt der Falterfigur mit dem Punkt X=40 und Y=35 im XY-Koordinatensystem zusammenfallen lassen wollen! Dann können wir den Griffel durch eine Anweisung LEER(40.0,35.0) positionieren. Wir könnten das gleiche durch die Anweisung LEER(A,B) erreichen, in der die Zahlenwerte durch die Variablen A und B ersetzt worden sind, wenn wir uns zuvor unzweideutig darauf geeinigt hätten, daß A den Wert 40 und B den Wert 35 hat. Wie bringt man eine solche Übereinkunft im Rahmen der Sprache G1 zustande? Nun, das Sprachmittel, das G1 in diesem Fall bereithält, ist die Wertzuweisung. Sie hat die Form

(1) name .= ausdruck

In (1) vertreten die Zeichenfolgen name und ausdruck bestimmte syntaktische Elemente der Sprache G1, man bezeichnet deshalb name und ausdruck auch als syntaktische Typen. Für name darf der Name jeder ALGOL-Variablen stehen, d. h. jede Zeichenfolge, die mit einem Buchstaben beginnt, mit Buchstaben oder Ziffern fortfährt und nicht mehr als sechs Zeichen enthält. Der syntaktische Typ ausdruck darf durch jede arithmetische Formel ersetzt werden, in der Namen, die Zeichen für die

vier Grundrechnungsarten und gewisse Funktionsausdrücke vorkommen. Die Bedeutung der Wertzuweisung (1) ist mit dem Exaktheitsprinzip im Einklang, denn (1) sagt jedem interpretierenden Automaten eindeutig: *Bis zur nächsten Wertzuweisung, in der ebenfalls der Name name links steht, vertritt die durch name benannte Variable, wo sie auch vorkommt, den Wert, den der Ausdruck ausdruck im Augenblick hat.* Wohlgemerkt ist (1) keine Gleichung, deshalb besteht das Wertzuweisungszeichen .= auch nicht einfach aus einem Gleichheitszeichen. Das leuchtet beispielsweise anhand der Wertzuweisung N.=N+1 ein, in der der Variablen N der Wert N+1 zugewiesen wird. Hat also N auf der rechten Seite von N.=N+1 den Wert 1, so hat N nach der Wertzuweisung den Wert 2. Mit Hilfe von Wertzuweisungen können deshalb auch Zählprozesse in Gang gesetzt werden.

Ein Programmbeispiel wird den Sinn der Wertzuweisung weiter klären:

(2)

```
 1 'BEGIN' 'REAL' A, B.,
 2                A.=40., B.=35., OPEN(0,0).,
 3  M1..          LEER(A,B).,
 4                LINE(A+25,B+15).,
 5                LINE(A+5,B+10).,
 6                LINE(A,B).,
 7                LINE(A−25,B+15).,
 8                LINE(A−5,B+10).,
 9  M2..          LINE(A,B).,
10                CLOSE
11 'END'
```

Wir betrachten zuerst die zweite Zeile von Programm (2). In ihr stehen die Wertzuweisungen A.=40 und B.=35, außerdem die in jedem Zeichenprogramm notwendige

Eröffnung. Nach Durchlaufung von Zeile 2 ist im Gedächtnis des interpretierenden Automaten gespeichert, daß A den Wert 40 und B den Wert 35 hat. Deshalb ist auch Zeile 3 (von der Zeichenfolge M1.. sehen wir zunächst ab) für ihn eindeutig verständlich: LEER(A,B) ist äquivalent mit LEER(40.0,35.0). Ebenso auch Zeile 4: LINE(A+25,B+15) ist äquivalent mit LINE(65.0,50.0). Die Programmzeilen 7, 8 und 9 benutzen die durch die Werte von A und B eindeutig gegebene Fußpunktlage der Falterfigur, deren rechter Flügel durch die Zeilen 4, 5 und 6 gezeichnet wurde, um auch den linken Flügel zu zeichnen. Es entsteht die vollständige Falterfigur in Bild 1 links unten, oberhalb des einzelnen rechten Flügels.

Nun kommen wir jedoch zum entscheidenden Vorteil von Programm (2) gegenüber dem Programmierstil von Programm (4/2.1.1): Durch geeignete Wertzuweisung in der zweiten Zeile kann der Falter an jeden beliebigen Punkt des Zeichenblatts kommandiert werden. Wir haben also durch die Abfassung der Programmzeilen 3 bis 9 mit Hilfe der Namen A, B eine vollständige Relativisierung der Lage der Falterfigur erreicht.

Programm (2) enthält einige Eigentümlichkeiten der Sprache G1, die noch zu erläutern sind. Auf das 'BEGIN'-Wortsymbol in Zeile 1 folgen die Zeichen 'REAL' A, B., In die Umgangssprache übersetzt bedeutet das Wortsymbol 'REAL': *Bis zum nächsten Strichpunkt folgt eine Serie von Namen von reellen Variablen.* Die ganze Zeichenfolge 'REAL' A, B., nennt man eine reelle Vereinbarung; in dem besonderen Fall der Zeile 1 von Programm (2) die Vereinbarung der reellen Namen A, B. Programmiersprachen mit Namensvereinbarungen, zu denen auch die Sprachen ALGOL und G1 gehören, haben den Vorteil, daß der interpretierende Automat gleich am Anfang des

Programms erfährt, mit welchen Variablen er im Programm zu rechnen und umzugehen hat. Ohne die Vereinbarungen müßte der Automat die Variablen durch einen komplizierten Analyseprozeß aus dem Programmtext heraussuchen, ein Unternehmen, das mit vielen Irrtumsmöglichkeiten behaftet ist.

Die Namen M1 und M2 in Programm (2) heißen Marken. Im Gegensatz zu den Nummern der Zeilenzählung, die wir links vom Programmtext angebracht haben, gehören die Marken zum Programm und sind deshalb Elemente der Sprache ALGOL. Die Zeichenfolgen M1.. und M2.. nennen wir Markenvereinbarungen. Zunächst dienen Markenvereinbarungen, zu denen jeder Name verwendet werden darf, der nicht irgendeine Variable im Programm benennt, zur Zeilenmarkierung. So haben wir die Marken M1 und M2 in Programm (2) gesetzt, um auf bequeme Art den lageunabhängigen Programmabschnitt herausheben zu können. Marken finden jedoch insbesondere als Sprungziele bei Programmverzweigungen Verwendung. In dieser wichtigen Funktion wurden Markenvereinbarungen zur Genese der Bilder 2 und 3 herangezogen, mehr darüber in den Abschnitten 2.1.4 und 2.1.5.

Dadurch, daß aus dem Abschnitt M1 bis M2 des Programms (2) alle überflüssige Zahleninformation entfernt und das Programm für beliebige geometrische Lage verallgemeinert wurde, begünstigt Programm (2) die Streuung der Falterfigur über das Zeichenpapier. Da wir uns, bevor wir in Abschnitt 2.2 die Zufallsgeneratoren besprechen und stochastische Dispersionsprozesse vorbereiten, im streng deterministischen Bereich bewegen, soll die Streuung der Falterfigur sehr einfachen multiplikativen Gesetzen folgen. Links oben in Bild 1 ist eine

deterministische Streuung der Falterfigur dargestellt. Wie wird sie durch eine einfache Verallgemeinerung von Programm (2) erzeugt?

2.1.3. Genese von Redundanz durch Laufanweisungen

Man kann die Faltersäule links in Bild 1 als translative Multiplikation der einfachen Falterfigur bezeichnen. Wenn wir nun die Sprachelemente von G1 mit der Umgangssprache verknüpfen, so können wir A den Wert 40 zuweisen und versuchsweise sagen: *Verändere B, mit dem Wert B=60 beginnend und in Schritten von 5.0 fortschreitend und mit B=150 endend. Führe bei jedem Schritt die Zeilen M1 bis M2 von Programm (2/2.1.2) aus!* Um dieses umgangssprachlich formulierte Programm und andere seiner Art in ALGOL ausdrücken zu können, wurde die Laufanweisung in die Sprache ALGOL aufgenommen und leistet nun auch den Konstruktionen der Erweiterungssprache G1 gute Dienste.

Der Satzanfang *„Verändere B, mit dem Wert B=60 beginnend und in Schritten von 5.0 fortschreitend und mit B=150 endend. Führe…"* läßt sich folgendermaßen in stark abgekürztes und symbolisiertes Englisch übersetzen:

(1) 'FOR' B.=60 'STEP' 5.0
'UNTIL' 150.0 'DO'

In (1) haben wir die englischen Wörter gleich als ALGOL-Wortsymbole geschrieben, so daß es uns leichtfallen wird, den durch (1) dargestellten Anweisungsteil als das multiplikative Element in dem ALGOL-Programm zum Zeichnen der Faltersäule wiederzufinden:

```
(2)   1  'BEGIN' 'REAL' A, B.,
      2              A.=40., OPEN(0,0).,
      3  L..        'FOR' B.=60 'STEP' 5.0
                     'UNTIL' 150.0 'DO'
      4  M1..       'BEGIN' LEER(A,B).,
      5                      LINE(A+25,B+15).,
      6                      LINE(A+5,B+10).,
      7                      LINE(A,B).,
      8                      LINE(A−25,B+15).,
      9                      LINE(A−5,B+10).,
     10  M2..                LINE(A,B)
     11              'END'.,
     12              CLOSE
     13  'END'
```

Die Marke L in Programm (2) deutet genau auf den Anweisungsteil (1). Dieser Anweisungsteil wirkt nun als multiplikativer Operator auf den Programmteil M1 bis M2, der uns schon aus dem Programm (2/2.1.2) als die Anweisungsfolge zum Zeichnen der einfachen Falterfigur bekannt ist.

Die Anweisungsfolge von L bis Zeile 11 heißt eine Laufanweisung. Ihre allgemeine Form ist

```
(3)   'FOR' wertzu 'STEP' arith1
      'UNTIL' arith2 'DO' laufkern
```

In (3) bedeutet wertzu eine Wertzuweisung, arith1 und arith2 bedeuten arithmetische Ausdrücke, d. h. Zahlen oder Variablen oder Zusammensetzungen aus solchen mit Hilfe der Grundrechnungsarten. Hinter 'DO' steht die als Laufkern bezeichnete Anweisung, hier vertreten durch den allgemeinen syntaktischen Typ laufkern. Die ganze Laufanweisung zerfällt deutlich in den multiplika-

tiven Operator von 'FOR' bis 'DO' (siehe (1)) und in den Operanden dieses Operators, anders ausgedrückt: Dasjenige was multipliziert werden soll. Dieses letztere ist der Laufkern. Ins Ästhetische und Zeichnerische übersetzt ist der Laufkern die zu vervielfachende Kernfigur, im Fall von Programm (2) die einfache Falterfigur.

Um dem Leser zu zeigen, welch mächtige Werkzeuge die Sprache ALGOL für die Genese von Graphiken bereithält, beschäftigen wir uns jetzt mit der Schachtelung von Laufanweisungen, bei der der Laufkern einer Laufanweisung selbst wieder aus einer Laufanweisung besteht. Wir erzeugen den Mittelteil von Bild 1, jenes rechteckige Feld aus rechten Falterflügeln durch eine Laufanweisung in einer Laufanweisung:

(4)

```
 1  'BEGIN' 'REAL' A, B., OPEN(0,0).,
 2          'FOR' A.=80 'STEP' 20.0
            'UNTIL' 120.0 'DO'
 3          'FOR' B.=10 'STEP' 10.0
            'UNTIL' 150.0 'DO'
 4          'BEGIN' LEER(A,B).,
 5                  LINE(A+25,B+15).,
 6                  LINE(A+5,B+10).,
 7                  LINE(A,B)
 8          'END'.,
 9           CLOSE
10  'END'
```

In Programm (4) besteht der innere Laufkern aus den Zeilen 4 bis 8. Da der Laufkern einer Laufanweisung selbst eine Anweisung ist, ergibt sich für uns die neue Erkenntnis, daß Anweisungen ihrerseits wieder aus Anweisungen zusammengesetzt sein können. In der Sprache ALGOL haben also auch einzelne Anweisungen

hierarchischen Aufbau. Man nennt eine Anweisung, die mit Hilfe von 'BEGIN' und 'END' mehrere Anweisungen zu einer Einheit zusammenklammert, eine Verbundanweisung. Der innere Laufkern aus den Zeilen 4 bis 8 des Programms (4) ist also eine Verbundanweisung. Diese Verbundanweisung ist abgesehen von OPEN(0,0) und CLOSE eine lagerelativisierte Form von Programm (4/2.1.1) und bewirkt in Programm (4) das Zeichnen des einzelnen Falterflügels der Flügelmatrix in Bild 1. Der Multiplikationsoperator, in ALGOL auch die Laufangabe genannt, der auf den inneren Laufkern wirkt, ist Doppelzeile 3. Die Zeilen 3 bis 8 jedoch bilden wiederum einen Laufkern. Dieser Laufkern besorgt, denn in ihm ist B die Laufvariable, das Zeichnen einer senkrechten Spalte der Flügelmatrix aus Bild 1. Nun fehlt nur noch der Multiplikationsoperator, der die Flügelspalte in drei Schritten von links nach rechts rückt. Dieser Operator steht in Doppelzeile 2. Die Flügelmatrix wird also, gesteuert durch die Laufangabe in Doppelzeile 2, spaltenweise gezeichnet; das Zeichnen der einzelnen Spalte wird gesteuert durch die Laufangabe in Doppelzeile 3.

2.1.4. Genesesteuerung durch Weichen

Die Laufanweisung ist also ein Redundanzgenerator, sie bewirkt die Vervielfachung von Grundfiguren. Im Programm selbst ist die Laufanweisung ein Mittel zur Informationsreduktion: Die in der Zeichnung oft erscheinende Grundfigur ist im Programm nur einmal durch den sie generierenden Laufkern vertreten.

Die Erkenntnis der sich selbst steuernden Prozesse hat die Kybernetik begründet. Da Programme Prozesse beschreiben, ist es selbstverständlich, daß die leistungsfähigen Programmiersprachen Mittel zur Selbststeuerung von Prozessen enthalten müssen. ALGOL stellt reiche

Mittel zur Selbststeuerung der Prozesse bereit, die durch ALGOL-Programme beschrieben werden. Diese Sprachmittel heißen Weichen, weil sie nämlich Prozesse nach der einen oder anderen Seite zu verzweigen vermögen.

So ziemlich der einfachste Weichentypus der Sprache ALGOL ist der folgende:

(1) 'IF' aussage 'THEN' anweisung1.,
anweisung2.,

In (1) sind aussage, anweisung1 und anweisung2 syntaktische Typen. Natürlich vertreten anweisung1 und anweisung2 beliebige Anweisungen. Der Typ aussage repräsentiert einen Abschnitt eines beliebigen ALGOL-Programms, der den Charakter einer Aussage, d. h. eines Satzes hat, der entweder wahr oder falsch ist. *Morgen regnet es* ist eine solche Aussage. Auch die Mathematik stellt Aussagen in großer Anzahl bereit, z. B. ist *3 ist kleiner als 2* eine Aussage, die ein falscher Satz ist.

In ALGOL werden Wortsymbole zur Bildung von Aussagen verwendet. So heißt 'LESS' *kleiner als* und 'NOTLESS' heißt *nicht kleiner als.* Ebenso leicht ins Deutsche übersetzbar sind 'EQUAL', 'GREATER', 'NOTEQUAL', 'NOTGREATER'. Nun können wir bereits ein Beispiel zum Weichenschema (1) bilden:

(2) 'IF' A1 'LESS' A2 'THEN' A3.=5.,
A4=6.,

Die Weiche (2) ist selbst eine Anweisung, jedoch eine Anweisung, die in Abhängigkeit des momentanen Werts der beiden Variablen A1 und A2 die eine oder die andere

von zwei weiteren Anweisungen auszuführen befiehlt. In die Umgangssprache übersetzt bedeutet (2) nämlich: *Ist die Aussage* A1 'LESS' A2 *wahr, dann führe erst* A3.=5 *aus, dann erst* A4.=6, *andernfalls führe sofort* A4.=6 *aus.* Entsprechend für das allgemeine Schema (1): *Ist die Aussage* aussage *wahr, dann führe erst* anweisung1 *aus, andernfalls führe sofort* anweisung2 *aus.* In vielen Weichen vom Typ (1) steht anweisung1 oder anweisung2 für einen besonders wichtigen Anweisungstyp, den wir noch nicht kennengelernt haben, nämlich für eine Sprunganweisung, auch 'GOTO'-Anweisung genannt. Die 'GOTO'-Anweisung hat die Form

(3) 'GOTO' marke

und befiehlt dem Automaten, der sie interpretiert, an der mit marke markierten Stelle des Programms fortzufahren. Z. B. könnte (2) folgendermaßen abgeändert werden:

(4) 'IF'A1 'LESS' A2 'THEN' 'GOTO'
'MARKE7., A4.=6.,

Umgangssprachlich bedeutet (4): *Falls A1 kleiner als A2 ist, gehe zur Marke* MARKE7, *sonst* A4.=6. Enthält also die Anweisung anweisung1 im Schema (1) eine Sprunganweisung, so kommt anweisung2 im allgemeinen nicht unmittelbar nach anweisung1 zur Ausführung.

Im Weichenschema (1) kann für anweisung1 — und nach Belieben selbstverständlich auch für anweisung2 — auch eine Verbundanweisung vom Typus 'BEGIN' ... 'END' eingesetzt werden. Ein Beispiel hierzu:

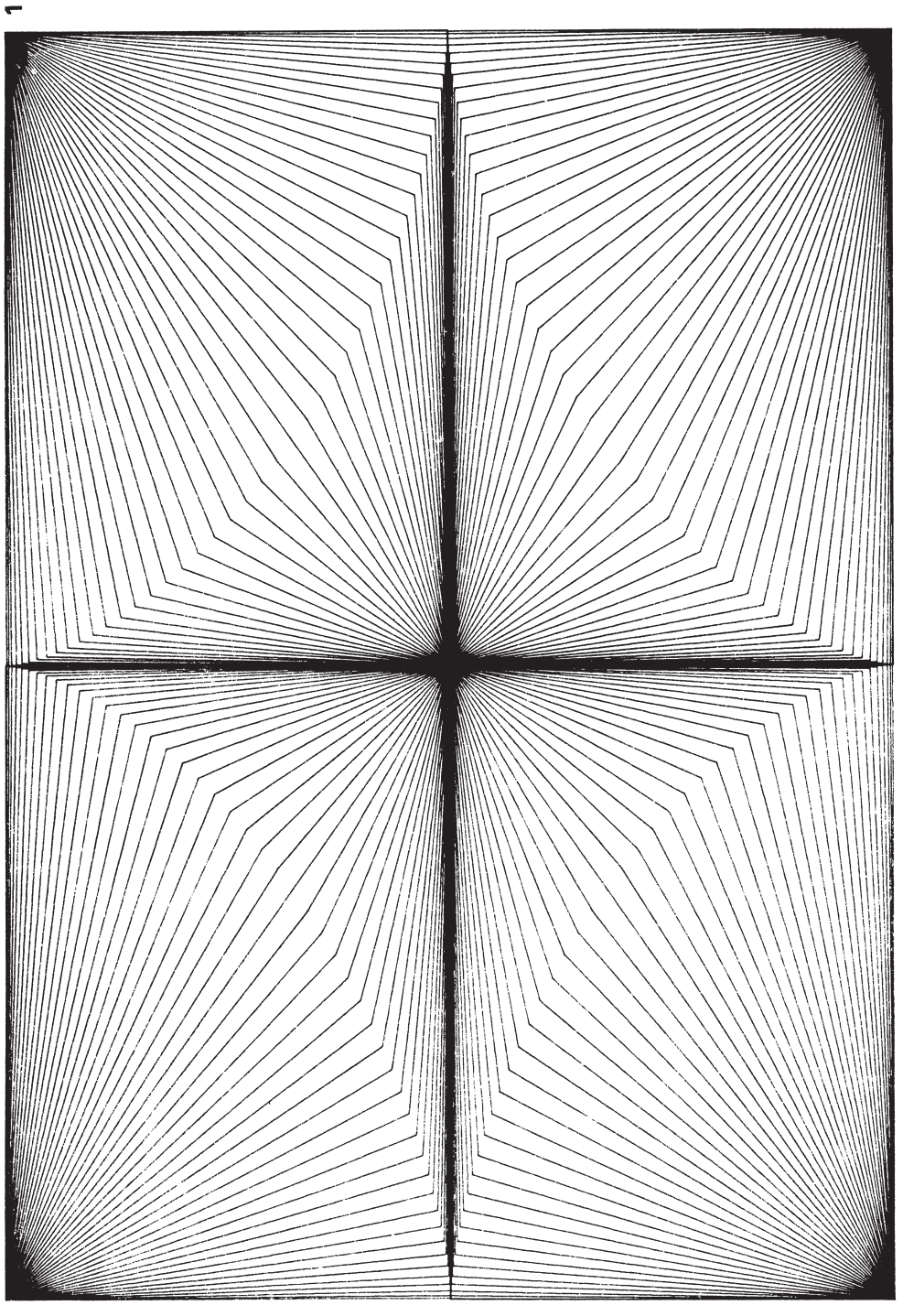
(5) 'IF' T 'NOTLESS' 300 'THEN'
(5a) 'BEGIN' LINE (A,— B)., 'GOTO' Z 'END'.,

(5b) 'IF' T 'NOTLESS' 200 'THEN'
(5c) 'BEGIN' LINE(−A,−B)., 'GOTO' Z 'END'.,

Vergleicht man (5) mit Schema (1), so tritt an die Stelle von aussage die Aussage T 'NOTLESS' 300, für anweisung1 tritt die ganze Zeile (5a) ein und für anweisung2 stehen die Zeilen (5b) und (5c). Sowohl anweisung1 als auch anweisung2 enthalten in diesem Fall Verbundanweisungen. Der interpretierende Automat reagiert dann, wenn T tatsächlich nicht kleiner als 300 ist, mit der Ausführung der Anweisung LINE(A,−B), anschließend fährt er mit der Anweisung fort, die er bei der Marke Z findet. Ist jedoch T momentan kleiner als 300, so wird Zeile (5a) übersprungen und die Zeilen (5b) und (5c) werden sofort ausgeführt. Bemerkenswert ist, daß anweisung2 im Fall (5) selbst eine Weiche ist.

Unsere Kenntnisse über Weichen und Programmverzweigungen benutzen wir nun dazu, ein Programm zur Genese von Bild 2 zu entwerfen. Als verschärfende Bedingung fordern wir, daß Bild 2 in einem Zug, also ohne die Verwendung einer LEER-Anweisung gezeichnet werden soll. Wir machen zuerst eine Voraussetzung über das Koordinatenkreuz, in das Bild 2 einzuzeichnen ist: Der Ursprung des XY-Koordinatensystems liege genau in der Mitte des querliegenden DIN-A 4-Zeichenbogens. Die rautenförmige Kurve, die im Bild sichtbar wird, genüge der Parameterdarstellung $X = A$ mal $\cos^3 T$, $Y = B$ mal $\sin^3 T$. Dabei ist T der laufende Parameter der Parameterdarstellung. Für das Original von Bild 2 wurde $A = 130$ mm und $B = 90$ mm gewählt. Wir betrachten T als ganzzahlige Variable, der wir die Werte 0 bis 399 zuordnen. Wird T als Winkel aufgefaßt, so teilt T den Vollkreis in 400 Neugrad ein. Kennzeichnet die Variable U das natürliche Winkelmaß, so kann man von T zu U leicht durch

Bild 2



die Wertzuweisung U.=T/400 * V übergehen, wenn V das Doppelte der Kreiszahl ist (das Sternchen * ist das in ALGOL verwendete Multiplikationszeichen). Führt man nun noch die leichtverständlichen und abkürzenden Wertzuweisungen C.=COS(U) und S.=SIN(U) ein, so wird durch die Anweisung LINE(A*C*C*C, B*S*S*S) eine Strecke von irgendeinem Punkt aus, z. B. vom Ursprung des Koordinatensystems, zur Bogenrautenkurve gezogen.

Nun wissen wir zunächst einmal, wie man den Griffel des Zeichenautomaten vom Nullpunkt des Koordinatensystems zur Bogenrautenkurve kommandiert. Bild 2 zeigt deutlich, daß diese Aktion bei der Genese des Bildes viele Male durchgeführt werden muß. Um auch den Rest der Bildgenese beschreiben zu können, führen wir eine Konvention ein, die wir im ganzen Bericht beibehalten wollen: Eine kleine arabische Ziffer z in einer Ecke des Bildes deute an, daß diese Ecke im Quadranten z des XY-Koordinatensystems liegt. Die Eins in Bild 2 bezeichnet also die Lage des ersten Quadranten des XY-Systems, dessen Ursprung mit der Bildmitte zusammenfällt. Diese Konvention macht jetzt die Programmierung des ersten Quadranten von Bild 1 ganz leicht. Aus der oben erläuterten Parameterdarstellung der Bogenrautenkurve entnehmen wir anschaulich, daß der lange Arm des in Bild 2 sichtbar werdenden Kreuzes die Länge A, der kurze die Länge B hat. Weist also der Griffel auf den Nullpunkt des XY-Systems, so zieht er durch die beiden Anweisungen

(6) LINE(A*C*C*C,B*S*S*S).,
LINE(A,B).,

eine Linie vom Nullpunkt zur Bogenrautenkurve und

von da zur rechten oberen, im ersten Quadranten liegenden, Ecke des Bildes. Die folgende Laufanweisung zeichnet den ganzen ersten Quadranten:

```
(7)   'FOR' T.=0 'STEP' 4 'UNTIL' 96 'DO'
(7a)  'BEGIN' U.=T/400*V.,
(7b)          C.=COS(U)., S.=SIN(U).,
(7c)          LINE(A*C*C*C,B*S*S*S).,
              LINE(A,B).,
(7d)          U.=(T+2)/400*V.,
(7e)          C.=COS(U)., S.=SIN(U).,
(7f)          LINE(A*C*C*C,B*S*S*S).,
              LINE(0,0)
(7g)  'END'
```

Der Multiplikationsoperator in Zeile (7) schaltet den Parameter T um jeweils vier Einheiten weiter. Im Laufkern selbst (Zeile (7a) bis (7g)) führt Doppelzeile (7c) vom Nullpunkt zur Bogenrautenkurve und von da zur rechten oberen Ecke, dann wird innerhalb des Laufkerns T um zwei Einheiten erhöht (Zeile (7d)) und Doppelzeile (7f) befördert den Griffel von der rechten oberen Ecke zur Bogenrautenkurve und von da zum Nullpunkt zurück. Dieses Pendelspiel zwischen dem Koordinatenursprung und der rechten oberen Ecke wiederholt sich, gesteuert vom Multiplikationsoperator, bis der ganze erste Quadrant gezeichnet ist, d. h. bis der Laufparameter T den Wert 96 erreicht hat.

Bis jetzt benötigten wir keine Weichen. Wie aber bringt man den Zeichenautomaten so weit, in den übrigen Quadranten die entsprechenden Ecken anzusteuern? Dabei soll die Triviallösung, nämlich die viermalige Niederschrift einer Laufanweisung von der Art (7) vermieden werden. Programm (8) zeigt die Lösung auf. In

diesem Programm ist die Laufanweisung (7) um drei Weichen vom Typus (1) erweitert worden. In der ersten dieser Weichen, in Zeile 10, wird gefragt, ob T nicht kleiner als 300 ist, anders ausgedrückt, ob T schon in den dritten Quadranten eingedrungen ist. Ist dies der Fall, so steuert der Weichenmechanismus Zeile 11 an und die Anweisung LINE(A,—B) dirigiert den Griffel wunschgemäß in die rechte untere Ecke. Anschließend führt die Sprungbedingung 'GOTO' Z den Programmablauf zur Marke Z, wo, genau ebenso wie in der Laufanweisung (7), das Rückpendeln der Zeichenlinie eingeleitet wird. Ist die Bedingung T 'NOTLESS' 300 jedoch nicht erfüllt, so steuert der Weichenmechanismus Zeile 12 an und es wird in einer neuen Weiche gefragt, ob T schon in den zweiten Quadranten eingedrungen ist. Natürlich ist die Weiche, die in Zeile 10 beginnt, genauso organisiert, wie die in Zeile 12 und ebenso, wie die in Zeile 14 beginnende. Die drei Weichen steuern in ihrem Zusammenspiel den Griffel jeweils in den Quadranten hinein, in dem ihn der Prozeß der Bildgenese gerade benötigt.

(8)

```
 1 'BEGIN' 'COMMENT' BOGENRAUTE.,
 2 'REAL' A, B, V, U, C, S.,
 3 'INTEGER' T.,
 4  OPEN(0,0)., A.=130., B.=90.,
 5  V.=2*3.14159.,
 6 'FOR' T.=0 'STEP' 4 'UNTIL'
   396 'DO'
 7 'BEGIN'  U.=T/400*V.,
 8            C.=COS(U)., S.=SIN(U).,
 9            LINE(A*C*C*C,B*S*S*S).,
10           'IF' T 'NOTLESS' 300 'THEN'
11           'BEGIN' LINE(A,—B).,
             'GOTO' Z 'END'.,
12           'IF' T 'NOTLESS' 200 'THEN'
```

```
13            'BEGIN' LINE(−A,−B).,
              'GOTO' Z 'END'.,
14            'IF' T 'NOTLESS' 100 'THEN'
15            'BEGIN' LINE(−A,B).,
              'GOTO' Z 'END'.,
16             LINE(A,B).,
17  Z..        U.=(T+2)/400*V.,
18             C.=COS(U)., S.=SIN(U).,
19             LINE(A*C*C*C,B*S*S*S).,
               LINE(0,0)
20  'END'.,
21  CLOSE
22  'END' BOGENRAUTE.,
```

Die durch die schrittweise Erhöhung von T im Kreis herumgedrehte Pendellinie durchläuft also gehorsam alle vier Quadranten des Koordinatensystems. Die Ansteuerung der Ecken ist ein Selbststeuerungsprozeß, schrittweise ausgelöst durch die Schwellenwerte 0, 100, 200 und 300, die der Parameter T nach und nach erreicht. Während der Bildgenese wird der Griffel nicht von der Zeichenfläche abgehoben.

Wir weisen zum Schluß dieses Abschnitts auf eine kleine Neuigkeit in Programm (8) hin: Der Kurvenparameter T durchläuft ganzzahlige Werte, während die Variablen A, B, V, U, C und S reelle, also auch nichtganzzahlige Werte annehmen. Da der Zeichenautomat reelle und ganzzahlige Größen auf unterschiedliche Weise in seinem Datengedächtnis abspeichert, muß ihm schon am Programmanfang die Reellität oder auch die Ganzzahligkeit der verschiedenen am Prozeß beteiligten Größen mitgeteilt werden. Für die reellen Variablen geschieht das durch die reelle Vereinbarung in Zeile 2, wie wir es in ähnlicher Weise schon bei Programm (2/2.1.2) kennenge-

lernt haben. Durch eine ganzzahlige Vereinbarung wird nun auch die ganzzahlige oder integre Größe T ins Programm eingeführt, siehe Zeile 3. In die Zeile 1 ist ein durch das Wortsymbol 'COMMENT' eingeleiteter Kommentar eingeflochten worden. Er steht außerhalb des Exaktheitsprinzips, sagt also nur dem menschlichen Interpreten etwas. Kommentare dürfen auch auf das 'END'-Symbol folgen (siehe Zeile 22).

2.1.5. Genesekonvergenz und Programmschleife

Wir haben in Abschnitt 2.1.3 die Laufanweisung studiert und haben sie dort auf die Genese der Faltersäule in Bild 1 links angewandt. Die Faltersäule kann als translative Multiplikation der einfachen Falterfigur betrachtet werden und die Laufanweisung, zerlegbar in einen Multiplikationsoperator und den Laufkern, bietet gerade die Hilfsmittel an, eine solche translative Multiplikation zu vollführen: Der Laufkern zeichnet die immer wiederkehrende Figur, der Multiplikationsoperator sorgt für die Wiederkehr. Im vorhergehenden Abschnitt 2.1.4 war der Laufkern für die Genese der einzelnen Pendellinie verantwortlich, der Multiplikationsoperator vervielfachte die Pendellinie über alle vier Quadranten des Koordinatensystems hin.

Im Zentrum steht hier der Begriff der Wiederkehr. Der Geneseprozeß läuft zyklisch ab, so wie vermutlich alle Prozesse, mit denen wir zu tun haben. Im Jahr kehren die Monate, Tage und Stunden ihrem Zeitmaß nach wieder, die Tätigkeit des Herzens ist zyklisch. Die meisten Prozesse, wie etwa der Jahreslauf, sind sogar mehrfach zyklisch organisiert, bei ihnen rotieren Zyklen innerhalb von Zyklen. Man kann deshalb auch umgekehrt sagen: Prozesse sind Hierarchien von Zyklen. Einen zweifach zyklischen Prozeß, beschrieben durch ein Programm mit

zwei ineinandergeschachtelten Laufanweisungen, haben wir sogar schon zur Genese einer Graphik benutzt, er generierte die Flügelmatrix in Bild 1.

Prozesse mit Anfang und Ende bedürfen einer Limitierung der sie konstituierenden Zyklen. Bei der Laufanweisung ist es das Wortsymbol 'UNTIL' und die Laufgrenze, die darauf folgt, was dem Zyklus ein gesteuertes Ende setzt. In der ALGOL-Theorie wird anstelle von *Zyklus* meistens das Wort *Schleife* benutzt. Laufanweisungen organisieren also limitierte Schleifen. Das auf sein Ende Zulaufen des Schleifenprozesses kann man auch als eine Konvergenz ansehen: Durch seine Beendigung wird das Generierte zum Ganzen.

Bemerkenswert ist nun, daß der Begriff der Programmschleife in ALGOL fundamentaler ist, als das Werkzeug der Laufanweisung. Die Laufanweisung ist nämlich ihrerseits redundant in ALGOL und reduzibel auf eine einfache Schleife. Bevor wir diese Aussage auch formal belegen, demonstrieren wir einen Geneseprozeß ohne Laufanweisung. Generiert wird Bild 3, das die Erscheinung der Prozeßkonvergenz deutlich anschaulich macht, wobei hier *Konvergenz* sogar mit dem Konvergenzbegriff der mathematischen Analysis übereinstimmt, denn in Bild 3 verschwindet ein Wiederkehrendes seiner Größe nach.

(1)

```
1  'BEGIN' 'COMMENT' SCHACHTELUNG.,
2  'REAL' LI, RE, UN, OB, H.,
3  H.=.5., OPEN(0,0).,
4  LI.=−130., RE.=130.,
5  UN.=−90., OB.=90.,
6  ANF..
7  LEER(0,OB)., LINE(LI,OB).,
```

1

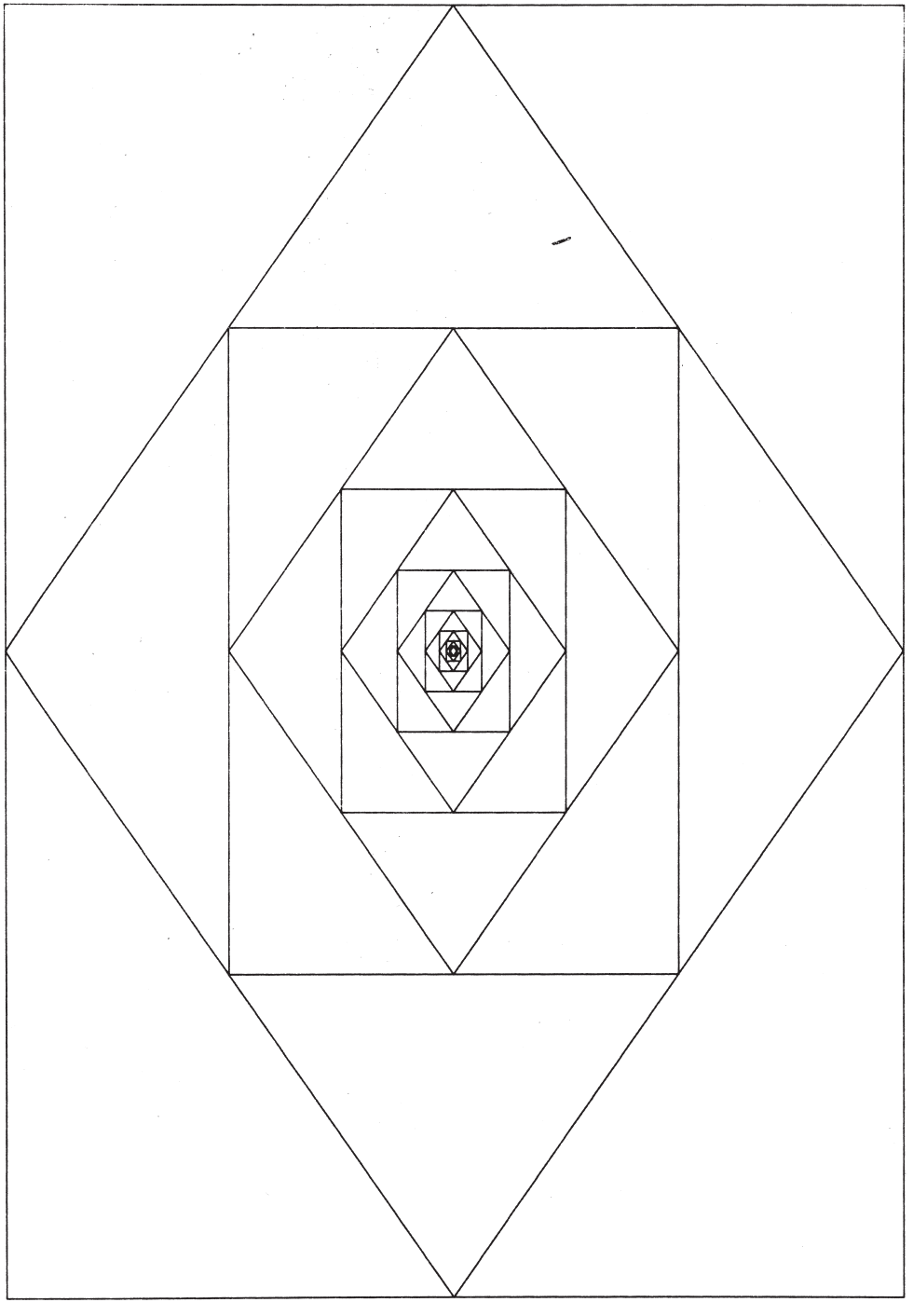


Bild 3

```
 8  LINE(LI,UN)., LINE(RE,UN).,
 9  LINE(RE,OB)., LINE(0,OB).,
10  LINE(LI,0)., LINE(0,UN).,
11  LINE(RE,0)., LINE(0,OB).,
12  LI.=LI*H., RE.=RE*H.,
13  UN.=UN*H., OB.=OB*H.,
14  'IF' RE 'GREATER' 1.0 'THEN' 'GOTO' ANF.,
15  CLOSE
16  'END' SCHACHTELUNG.,
```

Das Programm zur Erzeugung der Figur *Schachtelung* (siehe den Kommentar *Schachtelung* in Zeile 1 von Programm (1)) ist recht einfach gebaut: Die Koordinatenachsen verlaufen wieder durch die Bildmitte, die Figurgrenzen nach links und rechts sind durch die Werte der Variablen LI und RE, diejenigen nach unten und oben durch die Werte von UN und OB gegeben. Die Idee zur Konvergenzerzeugung beruht darauf, den Variablen LI, RE, UN und OB vor jedem Schleifendurchlauf neue verkleinerte Werte zuzuordnen, solange, bis die eingeschachtelte Figur unansehnlich klein geworden ist. Beim ersten Durchlauf haben LI und RE dem Betrag nach den Wert 130 Maßeinheiten (Millimeter im Original), UN und OB den entsprechenden Wert 90. Diese Werte erhalten die Variablen durch die Zuweisungen in den Zeilen 4 und 5. In Zeile 6 beginnt die Schleife. Sie veranlaßt den Automaten zunächst zum Zeichnen des ersten und äußersten Rechtecks und der ersten Raute. Danach werden in den Zeilen 12 und 13 die Größen LI, RE, UN und OB einfach halbiert — man beachte dazu, daß H aufgrund der dritten Zeile den Wert 0.5 = .5 hat. Anschließend fragt eine Weiche in Zeile 14, ob RE, d. h. die Auslegung des Rechtecks entlang der positiven X-Achse, noch größer als 1.0 sei. Nach dem ersten oder zweiten Weichendurchlauf ist das sicher noch der Fall, nicht aber

nach acht Durchläufen, wie man durch Abzählen der Rauten in Bild 3 feststellen kann. Ist jedoch die Weichenbedingung RE 'GREATER' 1.0 noch erfüllt, so steuert die Sprunganweisung 'GOTO' ANF den Programmablauf wieder zum Schleifenanfang in Zeile 6 zurück. Stellt aber die Weiche fest, daß die halbe breite Rechteckseite wirklich kleiner oder höchstens gleich 1 geworden ist, so leitet sie zur Ausführung von CLOSE über und der Prozeß endet in Zeile 16. Man versuche, sich den Zusammenhang zwischen der Schleifendynamik und der statischen Rautenschachtelung durch gedankliches Durchlaufen der Zeilen 6 bis 14 zu vergegenwärtigen.

Die Schleife in Programm (1) kann deshalb nicht in eine Laufanweisung übersetzt werden, weil der Programmentwerfer vor Ablauf des Programms gar nicht genau weiß, wieviel Schleifendurchläufe der technische interpretierende Automat tatsächlich durchzuführen haben wird. Hier ist der technisch realisierte Zeichenautomat dem Menschen insofern überlegen, als er auch einhundert oder mehr Schleifendurchläufe rasch und ohne Ermüdung absolvieren würde. Der Programmierer entwirft mit Hilfe der Werkzeuge Schleife und Weiche nur die Selbststeuerungseigenschaften des Programms und vertraut dem Exaktheitsprinzip. Hat nämlich der Programmierer sich so klar wie möglich ausgedrückt, so garantiert die innere Präzision der Sprache ALGOL, daß auch der Maschinenautomat das Programm in der vom Menschen intendierten Weise verstehen wird.

Nun wollen wir noch zeigen, daß sich zwar nicht jede Schleife durch eine Laufanweisung wiedergeben läßt, jedoch umgekehrt jede Laufanweisung mit Hilfe von 'GOTO'-Anweisungen in eine Schleife verwandelt werden kann. Das allgemeine Schema

(2) 'FOR' var.=ausdr1 'STEP' ausdr2
'UNTIL' ausdr3 'DO' kern

wobei var eine Variable ist, ausdr1, ausdr2 und ausdr3 arithmetische Ausdrücke bedeuten und kern den Laufkern vertritt, läßt sich folgendermaßen in die Schleifenform transformieren:

(3)

```
        var.=ausdruck1
M..     'IF' var-ausdr3 'GREATER' 0 'THEN'
        'GOTO' N.,
        kern.,
        var.=var+ausdr2.,
        'GOTO' M.,
N..
```

Man könnte den Programmabschnitt (3) sogar noch auf den Fall verallgemeinern, daß die Variable var negative Werte durchläuft, wir übergehen hier diese Weiterung.

2.1.6. Prozeduren in ALGOL und G1

In diesem Abschnitt besprechen wir die weitaus wichtigste Eigenschaft der Programmiersprache ALGOL, nämlich ihre Fähigkeit, Prozedurhierarchien zu beschreiben. Unter einer Prozedur versteht man dabei einen verhältnismäßig selbständigen, autark funktionierenden Programmabschnitt. Da Prozeduren immer in sie umfassende Rahmenprogramme eingeordnet sind, bezeichnet man sie auch als Unterprogramme.

Wesentlich ist, daß Prozeduren in einem Rahmenprogramm nicht jedesmal, wenn sie auftauchen, in ihrem vollen Wortlaut angeschrieben werden müssen. Sie werden nur einmal, am Anfang des Rahmenprogramms von

der ersten bis zur letzten Zeile aufgeführt. An den Stellen jedoch, an denen sie vom Programm aktiviert werden sollen, steht nur ihr Name und einige Hilfsangaben über ihre momentane Funktionsweise. Man unterscheidet die beiden Weisen des Erscheinens von Prozeduren in Programmen als Vereinbarung und Aufruf. Der Aufruf einer Prozedur ist immer eine Anweisung. Übrigens haben wir bereits in Abschnitt 2.1.1 spezielle Prozeduren kennengelernt, nämlich OPEN, CLOSE, LEER und LINE. Das sind gerade diejenigen Prozeduren, durch die das im Augenblick von uns erläuterte Programmiersystem G1 zur Genese von Streckenkomplexen über ALGOL hinausgeht.

Wir stellten schon früher fest, daß der Mensch vom Standpunkt der Informationstheorie aus betrachtet ein Automat ist. Nach dem Äquivalenzprinzip ist dieser Automat begrifflich gleichbedeutend mit einem Programm. Auch in dem Programm des menschlichen Verhaltens kann man Prozeduren und insbesondere die Funktionsteilung von Prozedurvereinbarung und Prozeduraufruf beobachten.

Allgemein ausgedrückt, ist jedes Erlernen von Praktiken Prozedurvereinbarung, während das Ausüben von Praktiken Prozeduraufruf ist. Beispielsweise erlernt der Buchbinderlehrling das Buchbinden. Die Fähigkeit, ein Buch zunftgemäß einzubinden, ist eine vereinbarte Prozedur, und ein Aufruf dieser Prozedur liegt dann vor, wenn der Geselle die Anweisung bekommt: *Binde dieses Buch ein!*

Die Bedeutung der Prozeduren für die Genese von Graphiken besteht in ihrer Fähigkeit, Grundelemente oder Grundbausteine von Graphiken darzustellen. Hier ent-

sprechen einander die funktionelle Autarkie der Prozedur einerseits, eines Bausteins einer Graphik andererseits. Besonders häufig wird man Prozeduraufrufe als Laufkerne von Programmschleifen und Laufanweisungen finden. Dabei kommt der kategoriale Charakter der Prozedur besonders klar zum Vorschein: Durch den Vervielfachungsprozeß der Schleife wird der Gehalt der Prozedur an den adäquaten Plätzen der Graphik individualisiert. Wir setzen diese ästhetischen Betrachtungen in einem besonderen Abschnitt fort und erläutern jetzt die Programmiertheorie des Prozedurbegriffs.

Betrachten wir ein Beispiel! Es sei ein vollständiger Falter, wie er in Bild 1 z. B. links unten erscheint, mit Hilfe einer Prozedur darzustellen. Ist diese Aufgabe gelöst, so ist die pfeilartige Faltersäule rechts in Bild 1 mit Hilfe wiederholter Aufrufe der erwähnten Prozedur zu generieren. Das Gesamtproblem ist also als Folge von zwei Teilaufgaben gegeben, wobei die erste Teilaufgabe die Vereinbarung einer Prozedur, die zweite Teilaufgabe den Aufruf der gleichen Prozedur verlangt. Wir schreiben die Problemlösung als Ganzes an und analysieren sie dann.

(1)

```
1  'BEGIN' 'REAL' P. Q.,
2          'PROCEDURE'
           FALTER(A,B,FAKTOR).,
3            'VALUE'  A, B, FAKTOR.,
4            'REAL'   A, B, FAKTOR.,
5            'BEGIN'  LEER(A,B).,
6                     LINE(A+25*FAKTOR,
                      B+15*FAKTOR).,
7                     LINE(A+5*FAKTOR,
                      B+10*FAKTOR).,
8                     LINE(A,B).,
```

```
 9                    LINE(A−25∗FAKTOR,
                      B+15∗FAKTOR).,
10                    LINE(A−5∗FAKTOR,
                      B+10∗FAKTOR).,
11                    LINE(A,B)
12          'END' FALTER.,
13           P.=200., OPEN(0,0).,
14          'FOR' Q.=10 'STEP' 5.0
            'UNTIL' 135.0 'DO'
15           FALTER(P,Q,0.5+(Q−10)∗1.5/125).,
16           CLOSE
17 'END'
```

Offenbar muß die Formulierung der zweiten Teilaufgabe auf die Lösung der ersten zurückwirken, denn die Figur in Bild 1 rechts zeigt Falter veränderlicher Lage und wachsender Größe. Nun sind wir allerdings von früher her schon im Besitz einer Verbundanweisung, die einen einzelnen Falter in beliebiger Lage generieren kann. Diese Verbundanweisung erscheint in Programm (2/2.1.3) in den Zeilen 4 bis 11. Durch Multiplikation aller relativen Koordinaten in den LEER- und LINE-Anweisungen der Verbundanweisung mit einem konstanten Faktor kann man sie leicht in eine Verbundanweisung verwandeln, die einen Falter beliebiger Größe generiert. Die Zeilen 5 bis 12 in Programm (1) des vorliegenden Abschnitts zeigen die so gewonnene neue Verbundanweisung. Nun haben wir die erste Teilaufgabe, die in der Vereinbarung eines Unterprogramms zur Genese eines Falterflügels beliebiger Größe besteht, schon fast gelöst. Um verstehen zu können, warum noch einige Zeilen mehr vonnöten sind, um das Unterprogramm zu einer vollgültigen Prozedur im ALGOL-Sinn zu machen, müssen wir uns an das Exaktheitsprinzip erinnern. Das Exaktheitsprinzip fordert vollständige Verständlichkeit

eines Programms für alle Automaten, die es realisieren. Auch der Mensch, der eine ihm unbekannte Prozedur zu analysieren hat, muß eine ganze Reihe vorbereitender Daten über die betreffende Prozedur erfahren. Zu diesen Daten gehört ein Hinweis, der einen Programmabschnitt als Prozedur kennzeichnet und der auch den Namen der Prozedur verrät, unter dem sie später aufgerufen werden kann. Dieser Hinweis steht in Doppelzeile 2 von Programm (1):

(2) 'PROCEDURE' FALTER(A,B,FAKTOR)

Das Wortsymbol 'PROCEDURE' sagt: *Jetzt kommt eine Prozedur!* FALTER(A,B,FAKTOR) ist der Name der Prozedur in allgemeiner Form. Schon von der Besprechung der Prozeduren LEER und LINE in Abschnitt 2.1.1 her erfassen wir, daß A, B und FAKTOR die Parameter der Prozedur FALTER sind, und zwar an dieser Stelle die Formalparameter. Zeile 3 enthält weitere vorbereitende Daten zum Zweck der vollständigen Kennzeichnung der Prozedur. Hier steht die Wertaufrufliste, d. h. die Liste derjenigen Parameter, die durch Wertaufruf in die Prozedur eingebracht werden. Wir erklären diese Zeile erst im Zusammenhang mit dem Aufruf der Prozedur in Zeile 15 des Programms. Zeile 4 wird als die Spezifikation bezeichnet. Die Spezifikation der Prozedur führt alle Parameter auf und stellt ihnen jeweils den Typ voran, dem sie angehören. Da unsere Prozedur FALTER(A,B,FAKTOR) nur reelle Zahlen als Parameter hat, erscheint in der Spezifikation auch nur das Wortsymbol 'REAL'. Es sei jedoch vermerkt, daß als Prozedurparameter mannigfaltige und sehr unterschiedliche Größen auftreten können, nämlich neben ganzen und reellen Zahlen auch Vektoren und Matrizen, ferner Marken, Aussagenvariablen und sogar weitere Prozeduren. Dies möge andeu-

ten, welch verschiedenartige und den Bereich des formal Möglichen fast ausschöpfende Informationsarten vom Instrumentarium der Prozeduren erfaßt werden. Alle diese Informationsarten können selbstverständlich auch zum Zweck der Genese von Graphiken von den Prozeduren verarbeitet werden. Zeile 4 schließt die vorbereitenden Daten der Prozedur ab. Zusammenfassend nennt man die Zeilen 2 bis 4 den Kopf der Prozedurvereinbarung.

Auf den Kopf der Prozedurvereinbarung folgt der Prozedurrumpf. Kopf und Rumpf zusammen bilden die vollständige Prozedurvereinbarung. Der Prozedurrumpf ist das eigentliche Programmstück, das die Funktion der Prozedur beim späteren Aufruf festlegt. Im Fall der Prozedur FALTER reicht der Rumpf von Zeile 5 bis 12. Diese Zeilen haben wir schon weiter oben besprochen.

Wir nehmen nun an, das ganze Programm (1) stehe im Informationsspeicher des Automaten und wir vergegenwärtigen uns, wie der Automat das Programm Zeile für Zeile zum Zweck seiner schrittweisen Aktualisierung abtastet. In Zeile 1 erfährt der Automat, daß zwei reelle Variablen P und Q im Spiel sein werden und er hält zwei Speicherzellen bereit, in denen später die wechselnden Werte von P und Q Platz finden können. Doppelzeile 2 sagt dem Automaten, daß eine Prozedur mit dem Namen FALTER und drei Parametern A, B und FAKTOR als Baustein des Gesamtprogramms funktioniert. Für alle Parameter der Prozedur werden beim Aufruf Zahlenwerte eingesetzt werden (Zeile 3, zum Begriff des Wertaufrufs siehe später) und diese Werte werden reelle Zahlen sein (Spezifikation in Zeile 4). Wiederum kann der Automat aufgrund seiner Kenntnis des Prozedurkopfes, den er nun durchlaufen hat, drei Speicherzellen für die

Aufnahme der aktualen Parameter der Prozedur bereithalten. Die Zeilen 5 bis 12 teilen dem Automaten die Funktionsweise der Prozedur FALTER mit. Diese Zeilen enthalten den Prozedurrumpf, in dem die Formalparameter der Prozedur als die funktionierenden Variablen auftreten. Der Rumpf von FALTER ist sehr einfach gebaut, es sei angemerkt, daß jeder Prozedurrumpf in sich selbst eine beliebig komplizierte Hierarchie von anderen Prozedurvereinbarungen und Prozeduraufrufen enthalten darf.

Der Vereinbarungsteil des Programms (1) reicht von Zeile 1 bis 12. Solange der Automat diese Zeilen zur Kenntnis nimmt, bereitet er sich auf spätere Aktivität nur vor, er betätigt jedoch seine Effektoren noch nicht. Von Zeile 13 an beginnt der Automat das Programm auszuwerten, d. h. er setzt seine inneren Rechenwerke in Betätigung. Zunächst wird in Zeile 13 durch P.=200 die Abszissenlage der zu generierenden Falterpfeilfigur festgelegt, OPEN(0,0) ist die übliche Eröffnung. In Zeile 14 wird eine Laufanweisung zum Zweck der Wiederholung der einzelnen Falterfigur initiiert. Die Laufanweisung verändert die Ordinatenlage des Falters zwischen den Ordinaten Q=10 und Q=135 in Schritten von 5 Einheiten. Der Laufkern in Zeile 15 sei hier noch einmal angeschrieben:

(3) FALTER(P,Q, 0.5+(Q−10)*1.5/125)

Die drei Formalparameter A, B und FAKTOR sind jetzt durch die Aktualparameter P, Q und 0.5+(Q−10)*1.5/125 ersetzt worden. P ist während des Ablaufs der Laufanweisung eine Konstante, während Q sich bei jedem Durchlauf der Laufanweisungsschleife um 5 Einheiten ändert. Mit Q ändert sich auch der Aktualparameter, der für FAKTOR eintritt, und zwar hat er für Q=10 den Wert

0.5 und für Q=135 den Wert 2. Die einzelne Falterfigur wächst also, während sie entlang der Ordinate P=200 verschoben wird, ihre Größe weitet sich vom Einhalb- bis Zweifachen der Größe des *Urfalters* links unten in Bild 1. Die abschließenden Zeilen 16 und 17 des Programms (1) bereiten keine Verständnisschwierigkeiten.

Jetzt sind wir soweit, den genauen Sinn des Wortsymbols 'VALUE', das den Wertaufruf der auf es folgenden Parameter anzeigt, erklären zu können: Wird ein nach 'VALUE' stehender Formalparameter beim Prozeduraufruf durch einen arithmetischen Ausdruck ersetzt, wie es oben im Fall des Parameters FAKTOR tatsächlich geschieht, so wird der arithmetische Ausdruck vor seiner Verarbeitung durch den Prozedurrumpf erst ausgewertet, d. h. auf einen einfachen Zahlenwert reduziert. Im Gegensatz zum Wertaufruf steht der Namensaufruf, bei dem der arithmetische Ausdruck als Ganzes, d. h. als Rechenvorschrift die Stellen im Prozedurrumpf ersetzt, wo der korrespondierende Formalparameter erscheint. Auch Namensaufrufe sind wichtige Programmierwerkzeuge, sie sind sogar als höher organisiert zu betrachten als die Wertaufrufe.

Nach einigen Abänderungen könnte man Programm (1) leicht in den Rumpf einer noch höher organisierten Prozedur einbauen und so beispielsweise ein Unterprogramm gewinnen, das Falterpfeile beliebiger Größe an beliebigen Stellen des Koordinatensystems anbringt. Eine Prozedur nächsthöherer Stufe würde vielleicht Matrizen von Falterpfeilen generieren und so könnten wir durch Ineinanderschachtelung von Prozeduren zur Genese von immer komplizierter gebauten, hierarchisch geschichteten Graphiken gelangen, die alle auf der einfachen Grundfigur des Falters beruhen. Ziehen wir

einen Ausdruck von HELMAR FRANK[1]) heran so generiert eine bestimmte Prozedurhierarchie der Sprache G1 eine bestimmte Stufenfolge graphischer Superzeichen. Wir finden die bei FRANK (a.a.O., Seite 36) angegebenen Übergänge wieder: Der Birkhoffsche Übergang entspricht dem Einbau einer Prozedur in eine umfassende und der Molessche Übergang bedeutet das analysierende Aufsuchen einer Prozedur innerhalb einer zweiten.

An dieser Stelle schließen wir den Abschnitt 2.1 ab, der uns in die Genese von Streckenkomplexen eingeführt hat. Allein durch die vier Prozeduren LEER, LINE, OPEN und CLOSE ist uns ein riesiger Bereich graphischer Bilder erschlossen worden. Die Automaten, deren Verhalten vollständig in der Sprache ALGOL beschrieben werden kann, erfahren durch die Hinzufügung von LEER, LINE, OPEN und CLOSE eine Angliederung spezieller Effektoren, mit deren Hilfe ihnen die Freisetzung optischer Information gelingt. Wir bezeichneten die Erweiterung der Programmiersprache ALGOL um die vier graphischen Prozeduren als die Sprache G1. Jedes Programm der Sprache G1 entspricht nach dem Äquivalenzprinzip einem Zeichenautomaten, der eine Graphik aus geradlinigen Abschnitten generiert. Dabei kann die gleiche Graphik von verschiedenen Automaten generiert werden, die deswegen in eine Äquivalenzklasse fallen. Daher ist, wenn wir die erzeugenden Automaten einer Graphik ihre Generatoren nennen, die Klasse der Streckenkomplexe eindeutig abbildbar auf eine Klasse von Äquivalenzklassen jeweils untereinander äquivalenter Generatoren. Jeder Generator wiederum entspricht genau einem Programm der Sprache G1 und jedes solche Programm genau einem Generator.

---

[1]) H. FRANK: Kybernetische Analysen subjektiver Sachverhalte, Verlag Schnelle, Quickborn bei Hamburg 1964

2.2. Zufallsbehaftete Bildgeneratoren und die Sprache G2

Wir gehen nun einen Schritt über die Sprache G1 hinaus. Vielleicht macht ein Bildbeispiel auf die einfachste Weise deutlich, welcher neue Bereich der generativen Graphik uns dadurch erschlossen wird. Bild 4 zeigt einen Schwarm von hundert zufällig über die Bildfläche verteilten Faltern zufälliger Größe. Die Falter haben die gleiche Gestalt wie die aus Bild 1, tatsächlich sind sie alle durch Aufrufe der Prozedur FALTER aus Programm (1/2.1.6) generiert worden. Natürlich könnte Bild 4 auch durch ein Programm der Sprache G1 erzeugt werden, zu diesem Zweck müßte jedoch jeder einzelne Falter durch einen gesonderten Prozeduraufruf mit speziellen Aktualparametern von FALTER(A,B,FAKTOR) im Generator vertreten sein. Im Rahmen der Sprache G2 kann Bild 4 aber durch eine einzige Laufanweisung, in deren Kern die Prozedur Falter und zwei Zufallsgeneratoren erscheinen, hergestellt werden. Wir wollen jedoch schrittweise vorgehen und nicht sofort mit der Erläuterung des Generators von Bild 4 beginnen. Im nachfolgenden Abschnitt beschäftigen wir uns zunächst mit dem Begriff des Zufallsgenerators selbst. Zufallsgeneratoren sind, wie sich zeigen wird, nichts anderes als spezielle Prozeduren.

2.2.1. Zufallsgeneratoren

Das Verfahren, das wir zur Erzeugung von Zufallszahlen benutzen werden, ist außerordentlich einfach. Ist J1 die letzte aus einer Folge bereits erzeugter Zufallszahlen, so wird J1 mit der Zahl 5 multipliziert und das Ergebnis wird durch sukzessive Subtraktion absteigender Zweierpotenzen unter eine bestimmte Zahlenschranke herabgedrückt. Die so gewonnene Zahl setzt als neue Zufallszahl die schon früher erzeugte Folge fort. Will man z. B. ganzzahlige Zufallszahlen erzeugen, die kleiner als

Bild 4

128 sind, so kann man 127 als erste Zufallszahl wählen — oder auch als letzte Zufallszahl einer kürzlich erzeugten Folge — und multipliziert nun 127 mit 5. Dieses ergibt 635. Nun zieht man die Zweierpotenzen 512, 256 und 128 solange schrittweise von 635 ab, bis das Ergebnis wieder kleiner als 128 ist. Man erhält also zunächst 635 — 512 = 123, was aber bereits kleiner als 128 ist, so daß wir nichts weiter abzuziehen brauchen. Nun haben wir bereits zwei Zufallszahlen: 127 und 123. Beim nächsten Mal erhalten wir 123 * 5 = 615. Davon 512 abgezogen, ergibt 103. Wiederum beim nächsten Mal ist 103 * 5 — 512 = 3. Bis jetzt besitzen wir also die Zufallszahlen 127, 123, 103, 3. Von 3 * 5 = 15 braucht man überhaupt nichts abzuziehen, denn 15 ist schon kleiner als 128, also ist 15 die nächste Zufallszahl. Auch 15 * 5 = 75 ist kleiner als 128. Die Folge von Zufallszahlen ist also jetzt angewachsen auf 127, 123, 103, 3, 15, 75.

Setzt man das Verfahren fort, so ergeben sich noch einige neue Zufallszahlen, sehr bald jedoch wird die Folge von Zufallszahlen periodisch, wie wir später noch anschaulich machen werden. Die Periode besteht aus genau 32 Zahlen. Man bezeichnet die Algorithmen zur Erzeugung von Zufallszahlen als Zufallsgeneratoren. Zufallsgeneratoren von so kurzer Periode, wie sie der vorgeführte Algorithmus besitzt, sind jedoch für unsere Untersuchungen fast immer ungeeignet. Wir benötigen Zufallssubstrate, die jedenfalls in dem Bereich, in dem wir sie benutzen, außer dem reinen Zufall selbst keine Qualitäten zu den ästhetischen Gebilden beisteuern, zu deren Genese sie herangezogen werden. Das bedeutet praktisch nichts anderes, als daß die Zufallsgeneratoren eine so große Periode aufweisen müssen, daß sie sich für uns nicht bemerkbar macht. Erfreulicherweise ist das Verfahren, mit dem wir vorhin experimentierten, dann

durchaus brauchbar, wenn wir die Zweierpotenzen 128, 256, 512 durch die Potenzen 2 hoch 32 bis 2 hoch 34 ersetzen. In die Programmiersprache ALGOL übersetzt, wird der Zufallsgenerator durch die Zeilen 6 bis 15 des folgenden Programms wiedergegeben:

```
(1)  2  'BEGIN' 'COMMENT' ZUFALLSFOLGE 1.,
     3  'INTEGER' JI, JS.,
     4  'REAL' JA, JE.,
     5  'REAL' 'PROCEDURE' J.,
     6  'BEGIN'  'COMMENT' ZUFALLS-
                    GENERATOR J.,
     7              JI.=5*JI.,
     8              'IF' JI 'NOTLESS' 8589934592
     9              'THEN' JI.=JI-8589934592.,
    10              'IF' JI 'NOTLESS' 4294967296
    11              'THEN' JI.=JI-4294967296.,
    12              'IF' JI 'NOTLESS' 2147483648
    13              'THEN' JI.=JI-2147483648.,
    14              J.=JI/2147483648*(JE-JA) + JA
    15  'END' ZUFALLSGENERATOR J.,
    16  JS.=1306859721.,
    17  'BEGIN' 'REAL' U.,
    18  JI.=JS., OPEN(0,0).,
    19  JA.=0., JE.=90.,
    20  'FOR' U.=0 'STEP' 2 'UNTIL' 258 'DO'
    21  'BEGIN'  LEER(U+.5,0).,
                 LINE(U+.5,J).,
    22           LEER(U+1.5,J).,
                 LINE(U+1.5,0)
    23  'END'.,  CLOSE
    24  'END'
    25  'END' ZUFALLSFOLGE 1.,
```


In Programm (1) wird der Zufallsgenerator J durch eine

Prozedur vereinbart. Allerdings ist J eine Prozedur von einer Art, wie wir sie jetzt zum ersten Male antreffen. Solche Prozeduren heißen Funktionsprozeduren und ihre Eigenart besteht darin, daß ihr Aufruf keine gesonderte Anweisung darstellt, sondern durch Setzung des Funktionsnamens innerhalb eines arithmetischen Ausdrucks vor sich geht. Tatsächlich sind wir den Aufrufen von zwei Funktionsprozeduren schon einmal begegnet, allerdings waren es Prozeduren, die zum festen Bestand der Sprache ALGOL gehören und deshalb nicht explizit vereinbart zu werden brauchen. Es handelte sich um die als Standardprozeduren bezeichneten Funktionen SIN und COS, die wir in Programm (8/2.1.4) verwendeten. Dort lieferte z. B. in Zeile 8 der Aufruf COS(U) innerhalb der Anweisung C.=COS(U) den Wert der Variablen C. Ebenso erhalten wir durch z. B. den Aufruf LINE(U +.5,J) automatisch zuerst einen Aufruf der Funktionsprozedur J, der die Ersetzung der Variablen J durch die Zufallszahl bewirkt, die der Zufallsgenerator J, ausgelöst durch seinen Aufruf, in diesem Augenblick generiert.

Es ist in der Sprache ALGOL Vorschrift, am Kopf der Vereinbarung einer Funktionsprozedur anzugeben, von welchem Typ der Wert ist, den die Funktionsprozedur bei ihrem Aufruf liefert. Zufallszahlen sind reelle Zahlen, also muß J als 'REAL' 'PROCEDURE' vereinbart werden. Funktionsprozeduren können jedoch auch ganze Zahlen oder auch Wahrheitswerte erzeugen, in diesem Fall treten die Wortsymbole 'INTEGER' bzw. 'BOOLEAN' an die Stelle von 'REAL'.

Wir verstehen die Funktionsweise des Zufallsgenerators J leichter, wenn wir sie im Rahmen eines Programms zur Genese einer Graphik beobachten. Betrachten wir also Programm (1). Zur Vereinbarung des Zufallsgenerators

ist nicht viel zu sagen. Er wiederholt genau die Technik des sukzessiven Subtrahierens von Zweierpotenzen, die wir zu Beginn dieses Abschnitts umgangssprachlich erläutert haben. Von Bedeutung ist jedoch folgendes: Innerhalb des Prozedurrumpfes von J erscheinen die Variablen JI, JA und JE. Diese Größen sind nicht innerhalb des Prozedurrumpfes vereinbart worden, sondern in einem Programmstück, das den Rumpf von außen umfaßt. Wir finden die Vereinbarung von JI, JA und JE in den Zeilen 3 und 4 des Programms. Allgemein bezeichnet man Variablen, die außerhalb eines Prozedurrumpfes vereinbart werden, als global im Prozedurrumpf. An dieser Stelle müssen wir noch einen anderen Begriff einführen, der eng mit dem Begriff der Globalität einer Variablen verknüpft ist: Wir meinen den Begriff des Blocks. Von der Verbundanweisung unterscheidet sich der Block dadurch, daß auf das eröffnende 'BEGIN'-Wortsymbol des Blocks die Vereinbarung von einer oder mehrerer Größen erfolgt. Alle diese Größen heißen lokal in dem Block, an dessen Anfang sie vereinbart werden. Die gleichen Größen heißen global in allen Blöcken, die dem ersten Block eingeschachtelt werden. Da das ganze ALGOL-Programm selbst einen Block darstellt, ist jede am Anfang des Programms vereinbarte Größe lokal im äußersten Block, jedoch global in eingeschachtelten Blöcken des Programms. In bezug auf die Begriffe lokal und global verhält sich nun jeder Prozedurrumpf wie ein Block, beinhalte er Vereinbarungen oder nicht. Deshalb sind JI, JA und JE global im Prozedurrumpf von J. Wir ergänzen den Begriff der Globalität noch durch einen etwas verallgemeinerten: Größen, die in einem Programm erscheinen, jedoch nicht in ihm vereinbart, sondern stillschweigend oder ausdrücklich als bekannt vorausgesetzt werden, nennen wir horizontglobal. Wir werden künftig ausdrücklich auf das Auftauchen von globa-

len und meistens auch auf das von horizontglobalen Größen hinweisen. Horizontglobal sind z. B. in Programm (8/2.1.4) die Prozeduren SIN und COS. Außerdem sind in allen Generatoren die Prozeduren OPEN, CLOSE, LEER und LINE horizontglobal.

Die Globalität von JI, JA und JE im Rumpf von J hat folgende Konsequenz für das Funktionieren der Funktionsprozedur J: Diesen Variablen müssen vor dem ersten Aufruf der Prozedur J Werte zugewiesen werden. Natürlich hat JI in Zeile 7 von Programm (1) genau die Bedeutung, die der Zahl 127 als erster Zufallszahl in der anfangs erzeugten Serie 127, 123, 103,. . . zukam, m. a. W. ist JI der Initialwert der zu generierenden Folge von Zufallszahlen. Im Fall von J muß JI immer eine ungerade zehnstellige Zahl und kleiner als 2 hoch 32 sein. Tatsächlich erhält JI in Zeile 18 den Wert von JS, der in Zeile 16 zu 1306859721 festgesetzt wurde.

In Programm (1) enden die Vereinbarungen erst in Zeile 15. In Zeile 16 wird die schon erwähnte Hilfsgröße JS mit einem geeigneten Wert versehen. Erst in Zeile 17 beginnt der eigentliche graphische Generator, der den Zufallsgenerator J verwendet. Wohlgemerkt sind alle von Zeile 2 bis 15 vereinbarten Größen global im eingeschachtelten Block, der sich von Zeile 17 bis 24 erstreckt. Dieser Block besitzt jedoch eine eigene lokale Hilfsgröße U. Die Funktion des Blocks ist die Genese des oberen *Spektrums* in Bild 5. In Zeile 18 des Blocks erhält JI seinen Anfangswert und das Zeichnen wird eröffnet. Zeile 19 weist den Größen JA und JE Werte zu. JA und JE haben die Bedeutung des Anfangs- bzw. Endpunktes des Zahlenintervalls, in das der Zufallsgenerator J seine Ergebnisse hineinstreut. Man beachte hierzu die Funktion der Zeile 14 im Rumpf des Zufallsgenerators! Da wir vertikale Bal-

ken zufälliger Länge erzeugen wollen (siehe dazu Bild 5), müssen wir den Balken eine sinnvolle Maximallänge zuschreiben. Die Festlegung der Maximallänge mit 90 Einheiten erreichen wir durch die beiden Anweisungen in Zeile 19. Nun bleibt nur noch wenig zu tun: Wir programmieren eine Laufanweisung, die uns eine Anzahl von Balken zufälliger Länge nacheinander generiert. Die Abszissenlage der Balken wird durch die sukzessiven Werte der Laufvariablen U gesteuert. Im Laufkern (Zeilen 21 bis 23) fährt LEER(U+.5,0) zunächst den Fußpunkt X=U+.5, Y=0 des ersten Balkens an, dann zeichnet LINE(U+.5,J) den Balken mit der zufälligen Länge J — hier haben wir den ersten Aufruf der Funktionsprozedur J — vor uns. Um den Zeicheneffektor möglichst wirtschaftlich zu nutzen, wird jetzt sofort der Scheitelpunkt des nächsten Balkens angesteuert, diese Positionierung besorgt der Aufruf LEER(U+1.5,J). Man vergegenwärtige sich, daß bei diesem zweiten Aufruf J einen ganz anderen Zufallswert innerhalb des Intervalls 0, 90 liefert als beim ersten Aufruf. LINE(U+1.5,0) schließlich zeichnet den zweiten Balken, beim Scheitelpunkt beginnend, beim Fußpunkt endend. Während eines einzelnen Durchlaufs der Schleife werden also jeweils zwei der insgesamt 150 Balken des Spektrums gezeichnet. Das Spektrum läßt keine Periodizität erkennen.

2.2.2. Der Falterschwarm

Von Bild 5, das eine anschauliche Einführung in den zeichnerischen Gebrauch der Zufallsgeneratoren gab, kehren wir zu Bild 4 zurück. Wie sieht der Generator des Falterschwarms aus?

Sehr oft reicht zum Zeichnen von Zufallsgraphiken ein einzelner Zufallsgenerator nicht aus. Der Grund dafür

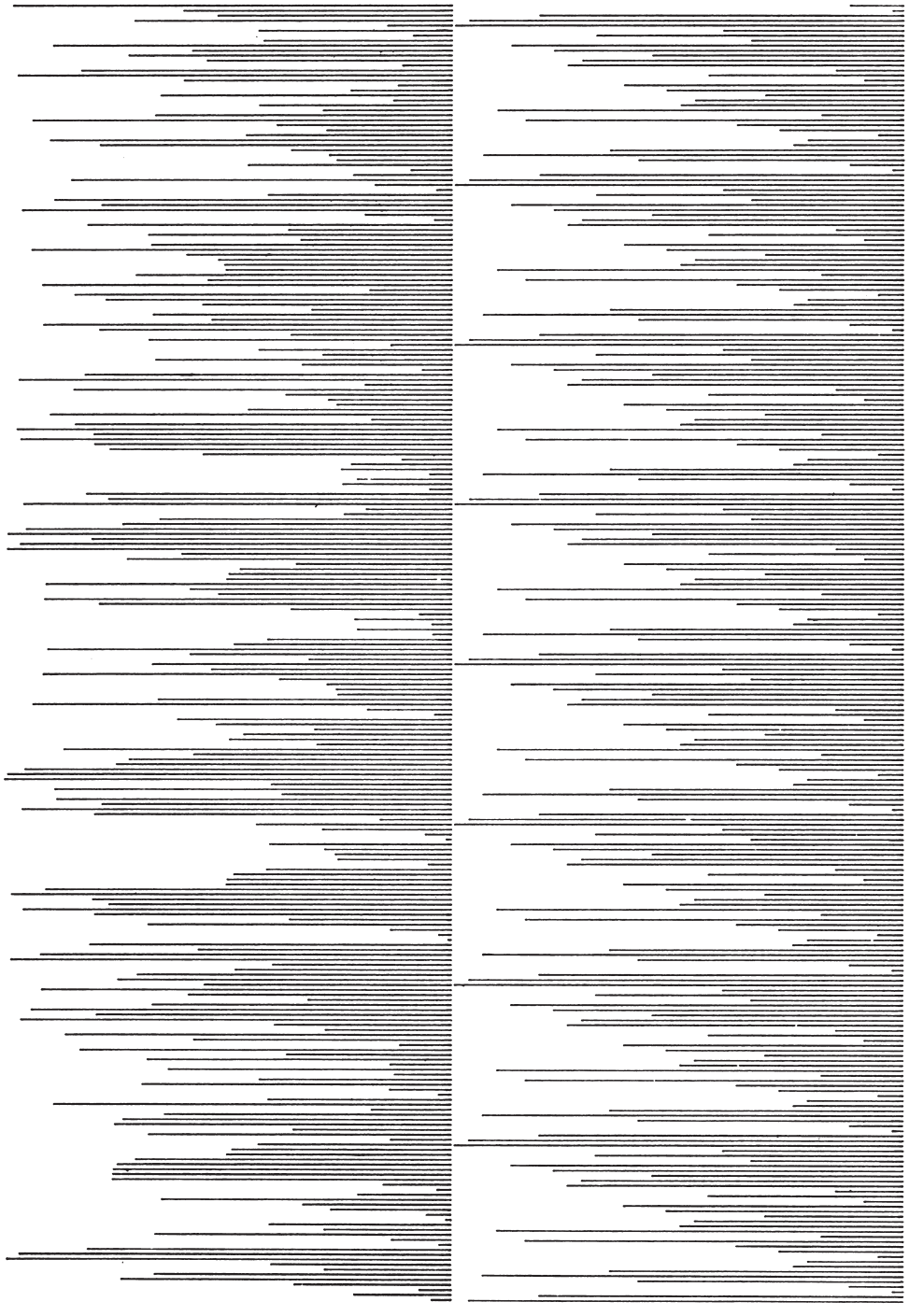


Bild 5

liegt darin, daß man verschiedene gestaltbestimmende Parameter einer Graphik gern durch verschiedene Zufallsgeneratoren regieren läßt, meistens benötigen die verschiedenen Parameter sogar verschiedene Streugrenzen. Außerdem vermindert die Verwendung von mehr als einem Zufallsgenerator die Gefahr des Einschleichens von Periodizitäten in die Struktur der Graphik. Wir führen deshalb eine Bezeichnungsweise ein, die es uns erlaubt, beliebig viele Zufallsgeneratoren in unseren Programmen aufzurufen: J1, J2, J3,... bezeichnen Zufallsgeneratoren, wobei die Vereinbarung der Prozedur Jn sich von den Zeilen 6 bis 15 des Programms (1/2.2.1) nur dadurch unterscheidet, daß beziehungsweise Jn, JIn, JAn, JEn an die Stelle von J, JI, JA, JE tritt. Die neuen Zufallsgeneratoren J1, J2, ... betrachten wir als horizontglobale Größen, d. h. wir rufen sie frei in den Programmen auf, sind uns jedoch bewußt, daß die Vereinbarungen von J1, J2, ... am Anfang solcher ALGOL-Programme stehen müssen, wenn sie tatsächlich in einer Datenverarbeitungsanlage zum Laufen gebracht werden sollen.

Nun sind wir in der Lage, die Sprache G2 zur Formulierung von zufallsbehafteten Bildgeneratoren durch eine Reihe von horizontglobalen Größen kennzeichnen zu können: G2 ist eine Erweiterung der Programmiersprache ALGOL um die Prozeduren OPEN, LEER, LINE, CLOSE, J1, J2, J3, ...

Vom mathematischen Standpunkt aus gesehen liefern die Zufallsgeneratoren J1, J2, ... gleichverteilte Zufallszahlen, d. h. die Wahrscheinlichkeit, daß ein Zufallswert in ein Teilintervall der Länge i von JA, JE fällt, ist immer gleich i/(JE—JA), unabhängig von der Lage des Teilintervalls innerhalb von JA, JE.

Wir kommen nun zum Thema dieses Abschnitts, nämlich zum Falterschwarm in Bild 4. Betrachten wir die Prozedur FALTER(A,B,FAKTOR) aus Abschnitt 2.1.6 und darüber hinaus eine reelle Hilfsgröße P und zwei ganzzahlige Größen JS1 und JS2 als horizontglobal, so wird der Generator des Falterschwarms ein verhältnismäßig kurzes Programm:

(1)

```
 1 'BEGIN' 'COMMENT'
   FALTERSCHWARM.,
 2 'REAL' R,S.,
 3 'INTEGER' I.,
 4 OPEN(0,0).,
 5 JI1.=JS1., JI2.=JS2.,
 6 JA1.=−130., JE1.=130.,
 7 JA2.=−90.,
 8 'FOR' I.=1 'STEP' 1 'UNTIL'
   100 'DO'
 9 'BEGIN'  P.=J1., R.=J1.,
10           S.=ABS(P−R).,
             JE2.=90−.3∗S.,
11           FALTER(.5∗(P+R),J2,.02∗S)
12 'END'.,
13 CLOSE
14 'END'     FALTERSCHWARM.,
```

in Zeile 5 erhalten die Initialgrößen JI1 und JI2 der Zufallsgeneratoren J1 und J2 die Werte von JS1 und JS2 zugewiesen. Dabei sind JS1 bis JS6 feste ganze Zahlen, die sich als Initialwerte für Zufallsserien eignen und die wir der Bequemlichkeit halber immer wieder in den Bildgeneratoren verwenden. Die Werte von JS1 bis JS6 sind durch die folgenden Anweisungen gegeben.

(2) JS1.=1306859721., JS2.=1485627309.,
JS3.=1649173265., JS4.=1805297143.,
JS5.=1973195467., JS6.=2013911545.,

Der Zufallsgenerator J1 bestimmt die Abszissenlage der Flügelspitzen der einzelnen Falterfigur. Deshalb kann das Abszissenintervall JA1, JE1 die gesamten 260 Millimeter ausmessen, die wir oft verwenden, um die lange Seite eines DIN-A 4-Zeichenblatts voll auszunützen. Nehmen wir den Koordinatenursprung in der Mitte des Zeichenblatts an, so sind JA1=−130, JE1=130, JA2=−90 brauchbare Werte. Der Zufallsgenerator J2 regiert, wie schnell zu erraten ist, die Ordinatenlage des Fußpunkts der einzelnen Falterfigur. Nun können wir allerdings leicht die untere Ordinatengrenze für den einzelnen Falter angeben, ihr Wert ist JA2=−90. Schwieriger ist jedoch die Bestimmung der Obergrenze. Man darf nicht einfach JE2=90 setzen, weil sonst die Flügelspitzen des Falters über den oberen Bildrand hinausgeraten können. In Programm (1) wurde folgender Weg zur Lösung dieses kleinen Problems beschritten: Innerhalb der Schleife, die alle Falter zeichnet (Zeilen 9 bis 12), werden in Zeile 9 zunächst die Abszissen P und R der Flügelspitzen des Falters durch zweimaligen Aufruf des Zufallsgenerators J1 festgelegt. In Zeile 10 ergibt S.=ABS(P−R), wobei die horizontalglobale Standardprozedur ABS den absoluten Betrag ihres Arguments liefert, die Flügelspannweite der einzelnen Falterfigur. Da .3∗S die Höhe der Falterfigur ist, genügt es, den Fußpunkt des Falters um mindestens .3∗S von der Maximalordinate 90 des Zeichenblatts nach unten abzurücken. Man erreicht dies dadurch, daß man die obere Intervallgrenze JE2 des Zufallsgenerators J2, der die Ordinatenlage des Falterfußes bestimmt, zu 90−.3∗S wählt, siehe Doppelzeile 10 von Programm (1). Als letztes Pro-

blem innerhalb der Schleife bleibt nun die korrekte Ersetzung der Formalparameter A, B und FAKTOR der Prozedur FALTER(A,B,FAKTOR). A bedeutet die Abszisse des Fußpunkts, man kann A also durch das arithmetische Mittel .5*(P+R) der Abszissen der Flügelspitzen ersetzen. Die Fußpunktordinate B ist durch den Wert des Zufallsgenerators J2 gegeben. Schließlich ist der Größenfaktor FAKTOR des einzelnen Falters genau ein Fünfzigstel der Flügelspannweite, was man daraus entnehmen kann, daß der *Normalfalter* aus Bild 1 den Größenfaktor 1 und eine Flügelspannweite von 50 Einheiten hatte. Es ist also FAKTOR durch .02*S zu ersetzen.

Zur Genese von Bild 4 wurden zwei Zufallsgeneratoren herangezogen. Der erste bestimmte auf dem Umweg über die Flügelspitzenabszissen die Abszisse des Fußpunkts des Falters, der zweite ergab die Ordinate des Fußpunkts. Natürlich könnte man unzählige Varianten des Falterschwarms generieren, z. B. ergäbe eine Limitierung der Flügelspannweite beispielsweise auf 60 Einheiten eine Ansicht des Schwarms gewissermaßen aus größerer Entfernung.

2.2.3. Entartete Zufallsgeneratoren

Es werde nun das Versprechen eingelöst, das wir am Anfang von Abschnitt 2.2.1 gegeben haben. Wir wollten nämlich die Folge von Zufallszahlen veranschaulichen, die wir durch Benutzung der Zweierpotenzen 128, 256, 512 erzeugten. Nichts liegt näher, als das obere Spektrum in Bild 5 noch einmal zu zeichnen, jetzt jedoch unter Verwendung der niedrigen Zweierpotenzen anstelle von 2 hoch 32 bis 2 hoch 34. Programm (1) generiert das untere Spektrum in Bild 5. Das Programm unterscheidet sich von Programm (1/2.2.1) nur in den Zweierpotenzen und

im Initialwert JS. Eine Zeile 1 wurde vorgeschaltet, die nach der Genese des oberen Spektrums den Griffel auf einen neuen Anfangspunkt in der linken unteren Ecke des Zeichenblatts verschiebt, in der Folge dort den Ursprung des Koordinatenkreuzes ansetzt und das Zeichen neu eröffnet.

(1)

```
 1  LEER(0,−90.0)., X.=0., Y.=0., OPEN(X,Y)..
 2  'BEGIN' 'COMMENT' ZUFALLSFOLGE 2.,
 3  'INTEGER' JI, JS.,
 4  'REAL' JA, JE.,
 5  'REAL' 'PROCEDURE' J.,
 6  'BEGIN' 'COMMENT'
                ZUFALLSGENERATOR J.,
 7              JI.=5*JI.,
 8              'IF' JI 'NOTLESS' 512
 9              'THEN' JI.=JI−512.,
10              'IF' JI 'NOTLESS' 256
11              'THEN' JI.=JI−256.,
12              'IF' JI 'NOTLESS' 128
13              'THEN' JI.=JI−128.,
14              J.=JI/128*(JE−JA) + JA
15  'END' ZUFALLSGENERATOR J.,
16  JS.=127.,
17  'BEGIN' 'REAL' U.,
18  JI.=JS., OPEN(0,0).,
19  JA.=0., JE.=90.,
20  'FOR' U.=0 'STEP' 2 'UNTIL' 258 'DO'
21  'BEGIN' LEER(U+.5,0)., LINE(U+.5,J).,
22              LEER(U+1.5,J)., LINE(U+1.5,0)
23  'END'.,     CLOSE
24  'END'
25  'END' ZUFALLSFOLGE 2.,
```

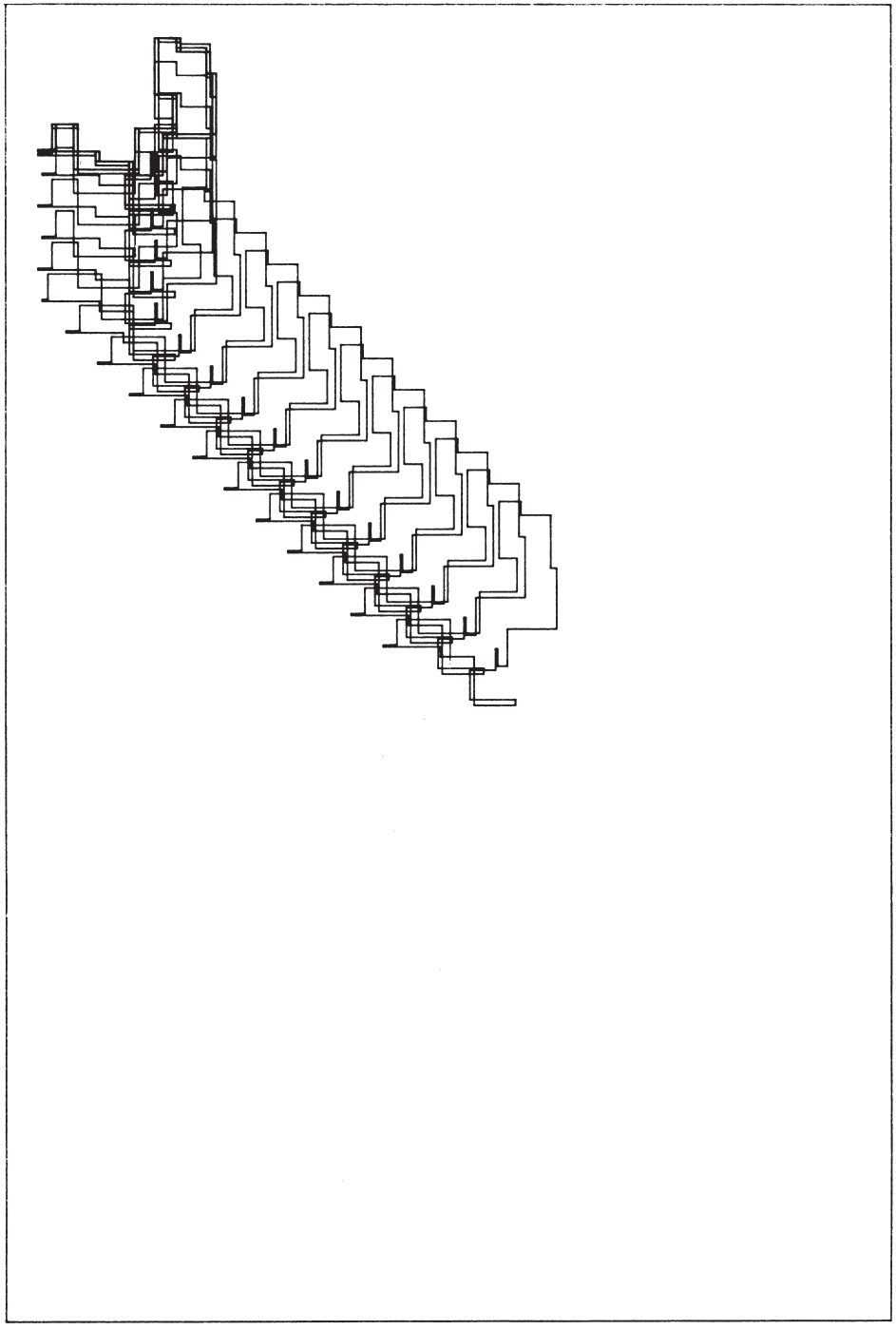

Das untere Spektrum enthüllt sofort die Periodizität der

Zahlenfolge 127, 123, 103, 3, 15, ... Allerdings wird als erster Balken nicht der zur Zahl 127 gehörige gezeichnet, sondern der zur Zahl 123, weil der Zufallsgenerator die Zahl 127 als Initialwert unter den Tisch fallen läßt.

Durch eine Serie von Bildern zeigen wir jetzt, wie sich Zufallsgeneratoren zu kurzer Periode, die wir auch als entartete Zufallsgeneratoren bezeichnen, auf die Genese von Graphiken auswirken. Wir stellen uns die Aufgabe, einen Irrweg zu generieren, der aus abwechselnd horizontalen und vertikalen Teilstrecken zufälliger Länge besteht. Die Tatsache, daß eine einzelne Teilstrecke sich bei waagrechtem Verlauf nach links oder rechts, bei senkrechtem nach oben oder unten wendet, lassen wir durch die Zufallsgeneratoren bestimmen. Wir fordern darüber hinaus, daß die Abmessung der Teilstrecken einen gewissen Höchstwert nicht überschreitet. Ferner postulieren wir, daß jede neu an den Irrweg angesetzte Teilstrecke um den gleichen Betrag von der Umrandung eines Rahmenrechtecks zurückgespiegelt wird, um den sie mit ihrem Endpunkt über diese Umrandung hinausgerät. Damit erreichen wir, daß der Irrweg sich innerhalb des Rahmenrechtecks bewegt. Als letzte Voraussetzung geben wir an: Der Irrweg soll genau in der Mitte des Rahmenrechtecks und mit horizontalem Verlauf beginnen; die Anzahl der Teilstrecken soll 2000 betragen.

Wir werden diese umgangssprachliche Beschreibung des Generators nicht in formales G2 übersetzen, die Übersetzung ist übrigens recht einfach. Der formale Generator benötigt zwei Zufallsgeneratoren: Einen Generator K1 für die Bestimmung der horizontalen, einen Generator K2 für die Bestimmung der vertikalen Teilstrecken. Die Zufallsgeneratoren K1 und K2 streuen ihre Erzeugnisse in ein Intervall hinein, das sich von −10 bis +10

Bild 6



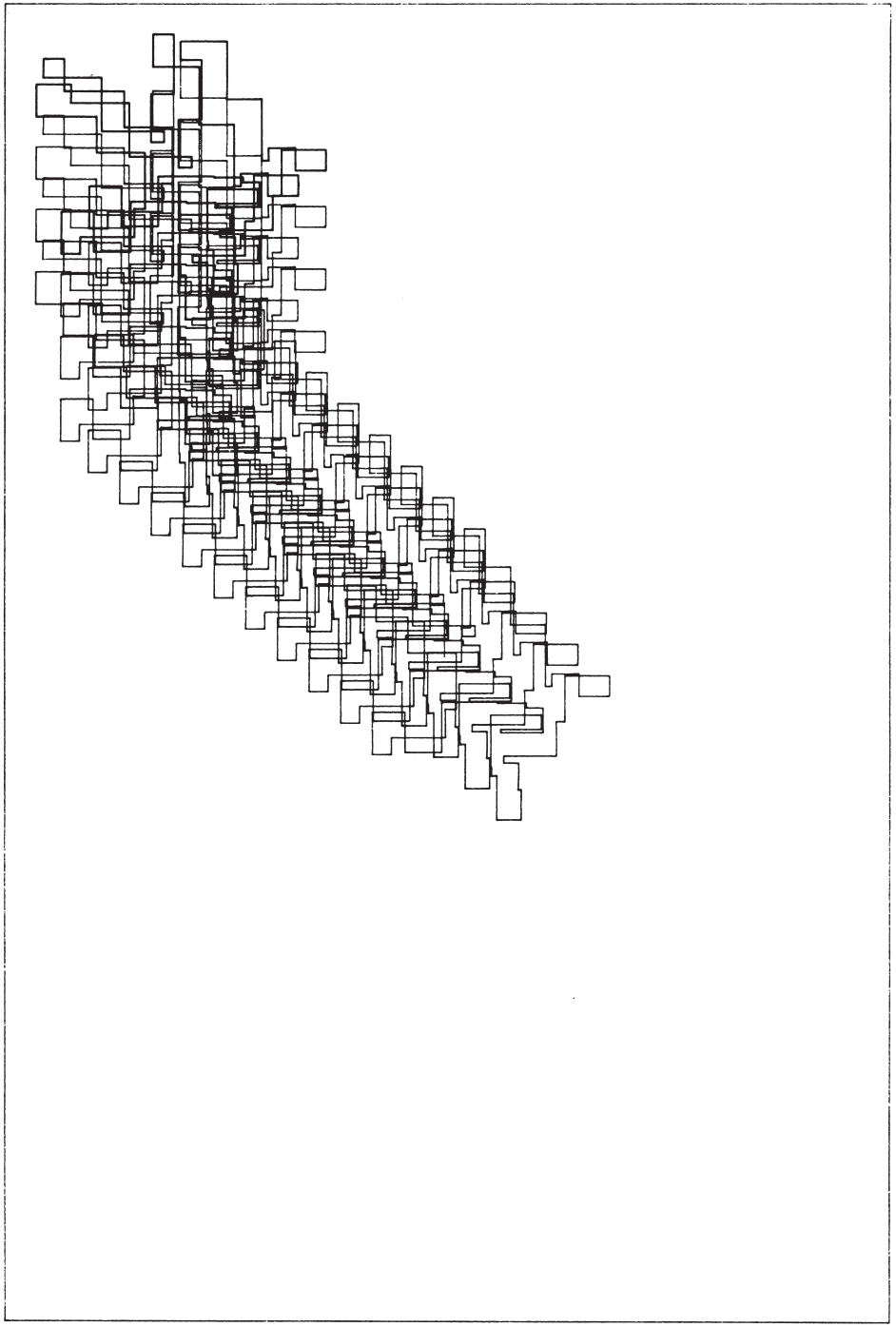
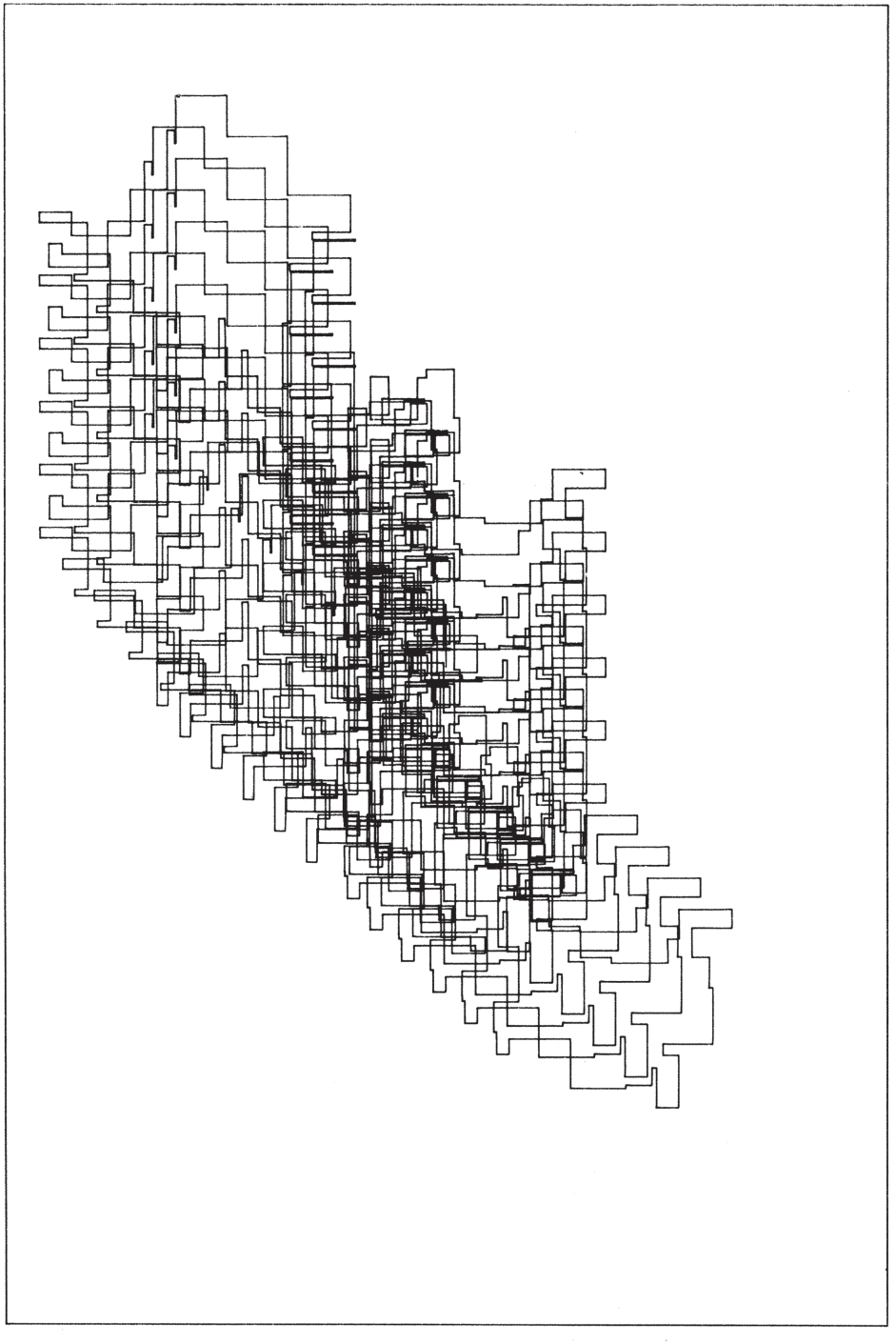


Bild 7

Bild 8



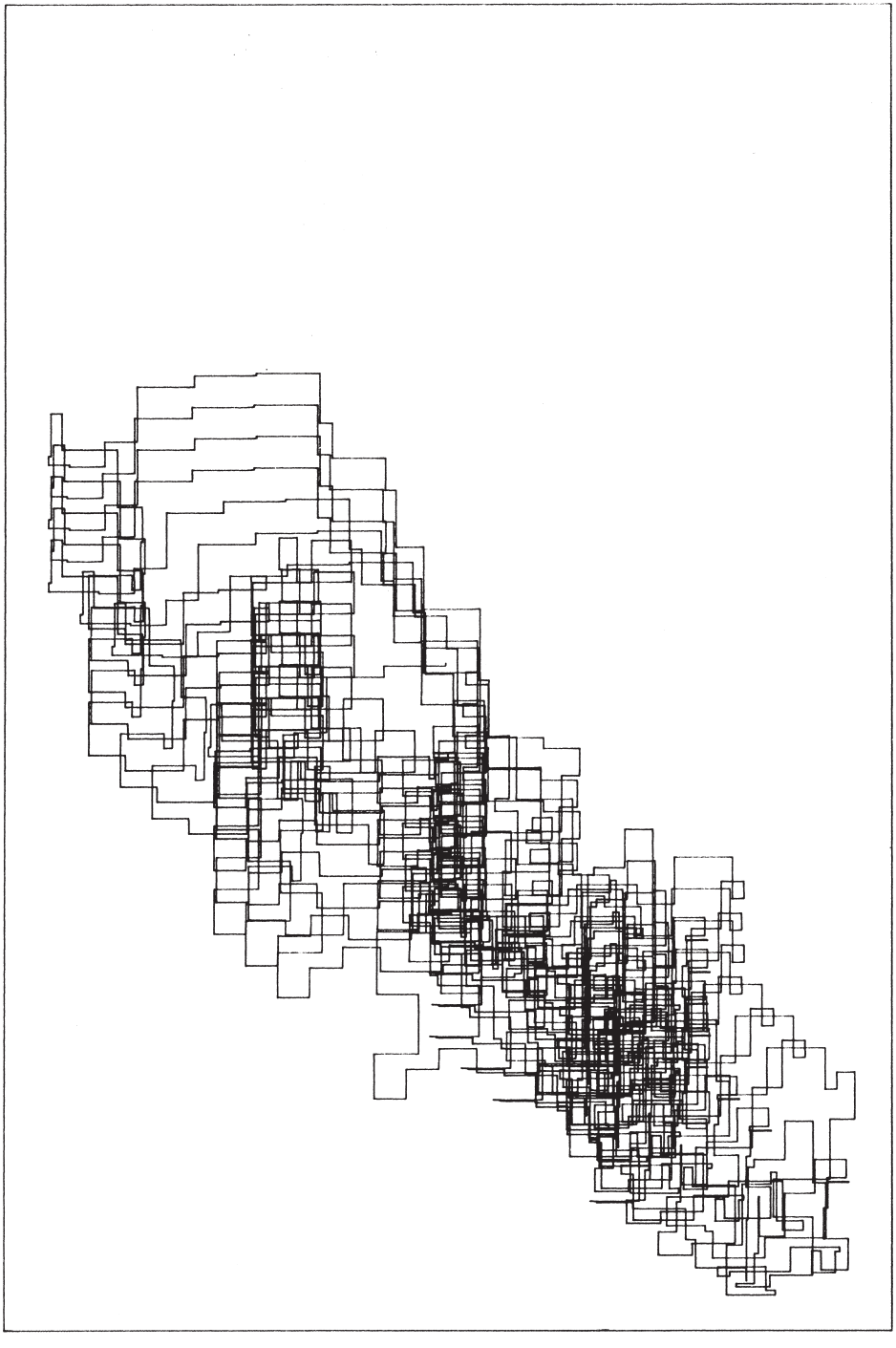


Bild 9

Bild 10

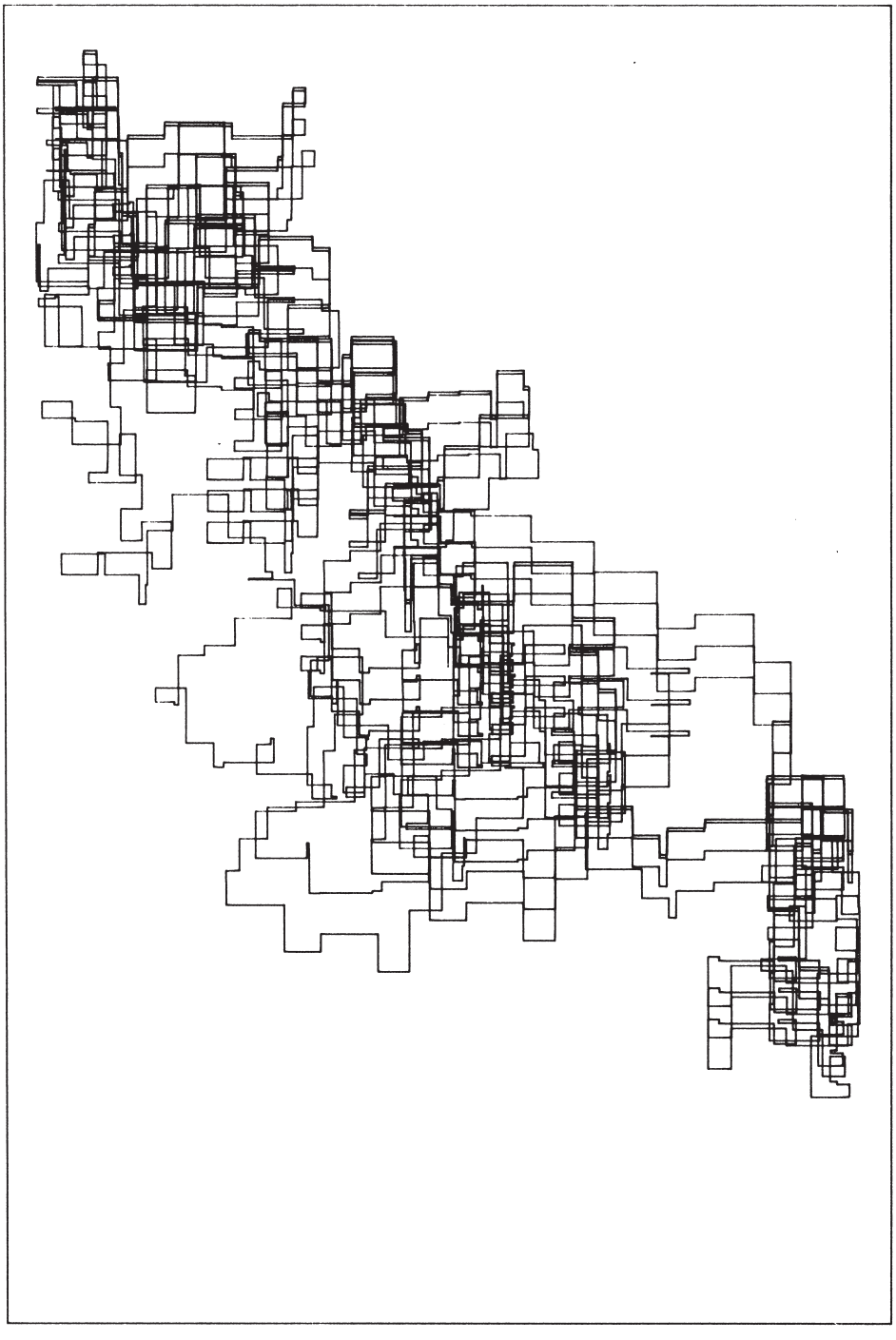
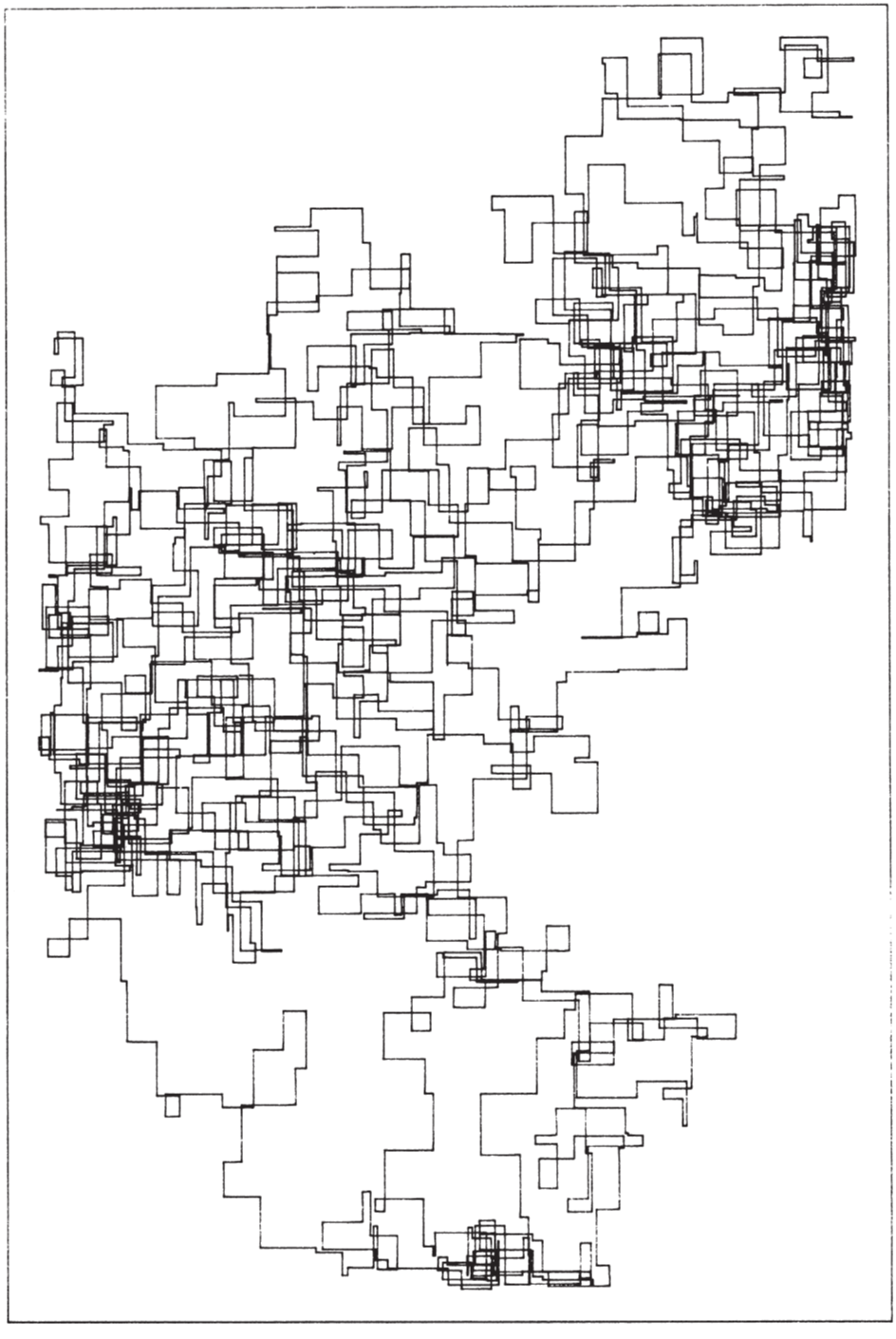


Bild 11

Bild 12



Einheiten erstreckt. Auf diese Weise wird auch entschieden, in welche Richtung der Irrweg sich jeweils wendet.

Unter Benutzung sukzessiver Tripel von Zweierpotenzen wurden sieben Bilder generiert. Der erste Tripel begann mit 64, das letzte mit 4096. Initialwert für den Zufallsgenerator K2 war jeweils die erste Zweierpotenz des Tripels abzüglich einer Eins. Initialwert für den Zufallsgenerator K1 war jeweils der Initialwert für K2 minus die Zahl 52. Man sieht, daß der Irrweg sich in Bild 6 nach rechts oben entfernt, wobei er schnell in einen Zyklus gerät. In der rechten oberen Ecke fängt der Irrweg sich durch wiederholte Reflexion am Rahmen und produziert nun eine Mehrfachüberdeckung des gleichen Streckenzuges. Diese Mehrfachüberdeckung tritt auch noch in Bild 7 auf, das Bild zeigt jedoch bereits vermehrten Merkmalsreichtum. Die Komplexität der Bildgestalt wächst nun von Bild zu Bild, bis man bei Betrachtung von Bild 11 den Eindruck gewinnt, daß eine individuelle Gestalt im wesentlichen zweimal vorhanden ist. Bei Bild 12 schließlich ist jede wahrnehmbare Einmischung der Periodizität der Zufallsgeneratoren K1 und K2 in die Bildindividualität verschwunden. Anhand dieses Bildes scheint der Aufweis gelungen zu sein, daß Zufallsgeneratoren, deren Tripel von Zweierpotenzen mit 2 hoch 32 beginnt, für die Zwecke der generativen Graphik ausreichen müssen, da doch schon die Zufallsgeneratoren K1 und K2 für Bild 12, deren Tripel mit 2 hoch 12 beginnt, sich so manierlich benehmen.

Der eigentümliche Reiz der Bilder 6 bis 11 legt die Vermutung nahe, daß entartete Zufallsgeneratoren von Bedeutung für Spezialuntersuchungen in der Computergraphik werden können.

Wir schließen den Abschnitt über die Sprache G2 ab. Zum Experimentieren mit Zufallsgraphik ist es übrigens nicht unbedingt notwendig, eine Datenverarbeitungsanlage zur Hand zu haben, man kann im Sinn des Äquivalenzprinzips Bildgeneratoren auch von Hand auswerten, muß dann freilich auch die Bildelemente selbst aufs Zeichenpapier aufbringen. In diesem Fall kann man einen Zufallsgenerator mit n-stelliger Genauigkeit leicht folgendermaßen realisieren: Auf n Pappkärtchen gleicher Größe schreibt man die Ziffern 0 bis 9. Die Kärtchen legt man dann in eine verschließbare Schachtel. Will man eine Zufallszahl im Intervall JA, JE erzeugen, so wiederholt man folgendes n-mal: Man zieht ein Kärtchen blind aus der Schachtel, notiert den Zahlenwert, legt das Kärtchen in die Schachtel zurück und schüttelt die Schachtel tüchtig. Nun dividiert man die so gewonnene n-stellige Zahl durch 10 hoch n, multipliziert das Ergebnis mit JE—JA und addiert schließlich JA (siehe auch Nees: Variationen von Figuren in der statistischen Grafik, Grundlagenstudien aus Kybernetik und Geisteswissenschaft, Quickborn bei Hamburg 1964).

Die Zufallsgeneratoren J1, J2, J3, . . . , die wir benutzen, gehen auf P. G. Behrenz zurück (Collected Algorithms from CACM, Algorithm 133, Random; New York 1962).

2.3. Kreisbogen und Rechteckfläche

In diesem Abschnitt ergänzen wir unser Sprachkonzept um die nützlichen Bildelemente Kreisbogen und Rechteckfläche. Der Kreisbogen ist so elementar, daß er durch eine horizontglobale Prozedur eingeführt werden muß, ganz ähnlich wie LEER und LINE. Im Gegensatz dazu kann die Rechteckfläche auf eine Folge von LEER- und LINE-Anweisungen zurückgeführt werden. Dabei wird die Rechteckfläche durch enge Aneinanderlagerung von

Strecken endlicher Strichdicke aufgebaut. Kreisbögen und Rechteckflächen werden uns im nächsten Kapitel gute Dienste leisten. Die Erweiterung der Sprache G2 um die Möglichkeit, Kreisbögen und Rechteckflächen zu benutzen, nennen wir G3.

2.3.1. Die Prozedur ZIRK

Ist der Zeichengriffel an einem Punkt X,Y angelangt, so kann er durch eine Anweisung LINE(P,Q) veranlaßt werden, eine Strecke zu zeichnen, die im Punkt P,Q endet. Durch die Anweisung LEER(P,Q) kann man ihn zum Punkt P,Q beordern, ohne daß er die Zeichenfläche berührt. Nun ist es denkbar, daß der Programmierer den Wunsch hat, den Punkt X,Y durch einen Kreisbogen mit dem Punkt P,Q zu verbinden. Dieser Kreisbogen bedarf allerdings weiterer Bestimmungsstücke, um eindeutig festgelegt werden zu können. Wir wählen folgende Möglichkeit:

(1) ZIRK(M,N,R,P,Q)

ist der Aufruf einer in einer Sprache G3 horizontglobalen Prozedur ZIRK. Es umfasse G3 die Sprache G2. In (1) sind M und N die Mittelpunktskoordinaten eines Kreises k mit dem Radius r, wobei r der absolute Betrag von R ist. Der Punkt P,Q muß auf dem Kreis k liegen. Ein Aufruf (1) der Prozedur ZIRK veranlaßt den Griffel, sich auf dem Kreis k vom augenblicklich erreichten Punkt X,Y zum Punkt P,Q im Uhrzeigersinn oder Gegenuhrzeigersinn zu bewegen, je nachdem R eine nichtnegative oder negative Zahl ist. Bei dieser Bewegung entsteht der gewünschte Kreisbogen. Fällt P,Q mit X,Y zusammen, so entsteht ein Vollkreis mit dem Radius R, wenn R positiv ist.

Ein Anwendungsbeispiel zeigt Bild 13. der Programmblock, durch den Bild 13 generiert wird, lautet folgendermaßen:

```
(2)   1 'BEGIN'
      2 'COMMENT'
        DEMONSTRATION KREISBOGEN.,
      3 'INTEGER' I., 'REAL' U, V, R.,
      4 OPEN(X,Y)., JI1.=JS1., JI2.=JS2.,
      5 JA1.=0., JE1.=130., JA2.=−80., JE2.=80.,
      6 'FOR' I.=1 'STEP' 1 'UNTIL' 200 'DO'
      7 'BEGIN' ANF..
      8 U.=J1., V.=J2.,
      9 'IF' ABS(V) 'LESS' 1.0 'THEN' 'GOTO' ANF.,
     10 R.=.5*(U*U+V*V)/V.,
     11 LEER(U,V)., ZIRK(0.0,R,R,−U,V)
     12 'END'.,
     13 CLOSE
     14 'END' DEMONSTRATION KREISBOGEN.,
```

Wie Zeile 11 des Programms zeigt, befindet sich der Aufruf von ZIRK im Innern der von Zeile 6 bis Zeile 12 sich erstreckenden Laufanweisung. Das bedeutet, daß insgesamt 200 Kreisbögen generiert werden (siehe Zeile 6). Der Anfangspunkt, an dem der einzelne Kreisbogen beginnt, ist U,V. Dabei werden die Koordinaten U und V des Anfangspunktes durch die Zufallsgeneratoren J1 und J2 in Zeile 8 berechnet. Zeile 9 sorgt im Fall, daß die Koordinate V einen sehr kleinen Wert hat, für die Ermittlung eines brauchbaren Anfangspunktes. Ist der Anfangspunkt endgültig festgelegt, so berechnet Zeile 10 durch ein wenig Elementargeometrie den Radius eines Kreises, der durch U,V und den Koordinatenursprung verläuft. Der Radius dieses Kreises stimmt mit dem absoluten Betrag von R überein, sein Mittelpunkt hat (wieder-

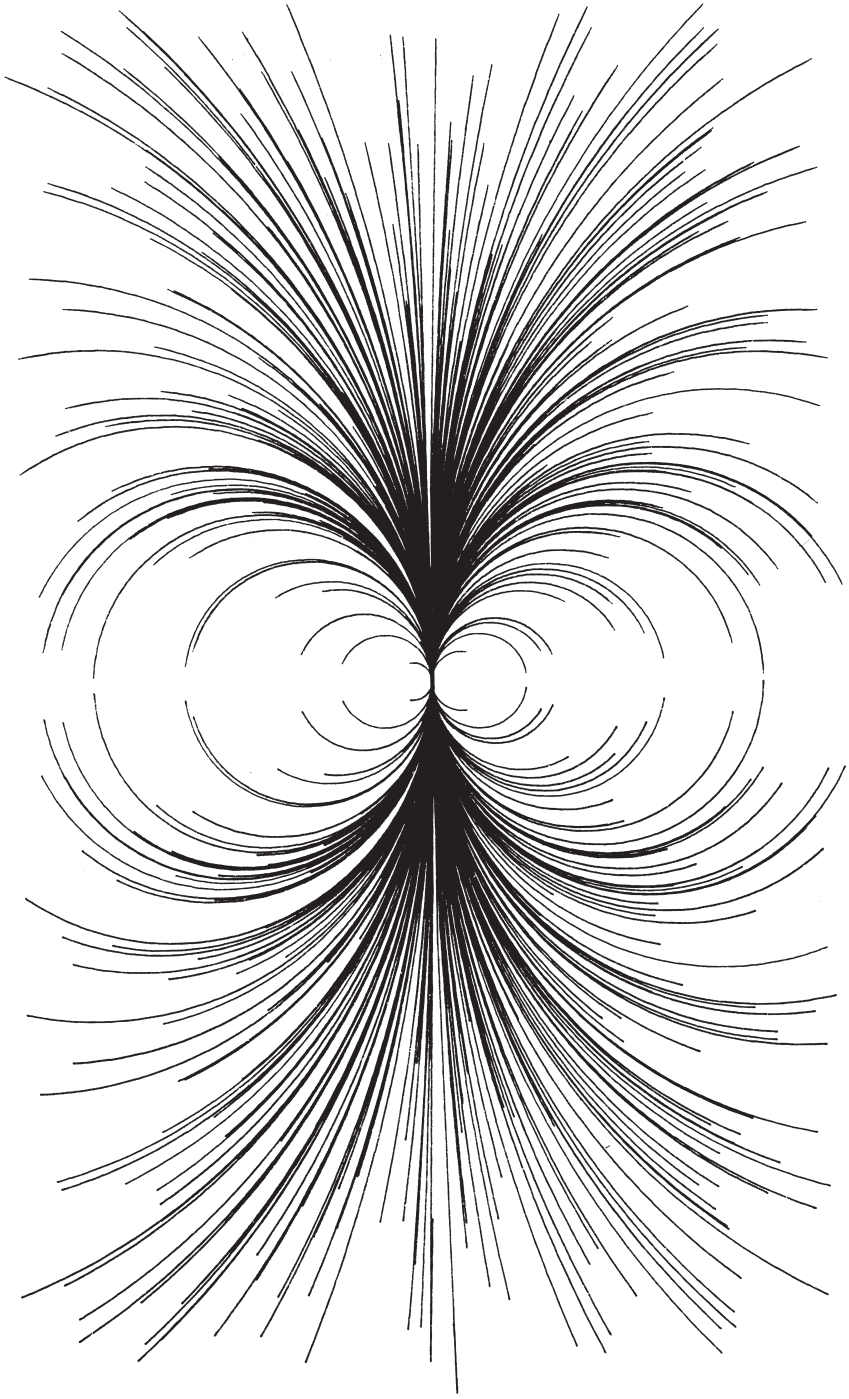


Bild 13

um mit Hilfe von Elementargeometrie) die Koordinaten X=0.0 und Y=R. Zeile 11 positioniert den Zeichengriffel zunächst im Anfangspunkt U,V und zieht dann einen Kreisbogen auf dem erwähnten Kreis bis zum Endpunkt −U,V. Der positive oder negative Wert von R legt den Kreisbogen in die obere oder untere Halbebene.

2.3.2.
Die Prozedur REC

Bild 14 zeigt eine Anordnung von Rechtecken. Wie wir schon in der Einleitung andeuteten, wird jedes der Rechtecke aus eng nebeneinanderliegenden Strecken aufgebaut. Dabei muß natürlich die Strichstärke des verwendeten Griffels berücksichtigt werden. Wir bezeichnen die Strichstärke mit S und führen sie als Parameter in eine Prozedur REC ein, die eine Rechteckfläche generiert. Der Aufruf von REC lautet

(1) REC(D,E,F,G,S)

Leicht zu erraten ist die Bedeutung der Parameter D bis G: Es sind D und E die Abszissen der vertikalen Kanten, F und G die Ordinaten der horizontalen Kanten der generierten Rechteckfläche. Wir analysieren nun den Rumpf der Prozedur REC und werden dabei Gelegenheit haben, die Verwendung von Wahrheitswerten — auch Boolesche Größen genannt — in Programmen zu studieren.

(2)

```
1  'PROCEDURE' REC(D,E,F,G,S).,
2  'VALUE' D, E, F, G, S., 'REAL' D, E, F, G, S.,
3  'BEGIN'  'REAL' A, H, L, Z.,
4                'BOOLEAN' B.,
5  A.=S−.1.,
6  B.='TRUE'.,
```

Bild 14

```
 7  'IF' ABS(E-D) 'LESS' ABS(G-F)
    'THEN' 'GOTO' FG.,
 8  H.=SIGN(G-F)*A., L.=F.,
 9  LEER(E,F)., LINE(E,G)., LINE(D,G)., LINE(D,F).,
10  SDE..
11  'IF' B 'THEN' 'GOTO' SDEB.,
12  Z.=D., 'GOTO' SDEF.,
13  SDEB.. Z.=E.,
14  SDEF.. LINE(Z,Y)., B.='NOT' B.,
15  'IF' ABS(G-L) 'LESS' A 'THEN' 'GOTO' EXIT.,
16  L.=L+H., LINE(X,L)., 'GOTO' SDE.,
17  FG.. H.=SIGN(E-D)*A., L.=D.,
18  LEER(D,G)., LINE(E,G)., LINE(E,F)., LINE(D,F).,
19  SFG..
20  'IF' B 'THEN' 'GOTO' SFGB.,
21  Z.=F., 'GOTO' SFGF.,
22  SFGB.. Z.=G.,
23  SFGF.. LINE(X,Z)., B.='NOT' B.,
24  'IF' ABS(E-L) 'LESS' A 'THEN' 'GOTO' EXIT.,
25  L.=L+H., LINE(L,Y)., 'GOTO' SFG.,
26  EXIT..
27  'END' REC.,
```

Die Boolesche Größe wird in Zeile 4 vereinbart. Sie erhält hier den willkürlich gewählten Namen B. Natürlich kann B, im Gegensatz zu ganzzahligen und reellen Größen, nur zwei Werte annehmen, denn eine Boolesche Größe kann ja nur entweder die Wahrheit oder die Falschheit einer Aussage kennzeichnen. In ALGOL und auch in der Sprache G3 bezeichnen wir den Wahrheitswert wahrer Aussagen mit 'TRUE' und den falscher Aussagen mit 'FALSE'. Mit einer Wertzuweisung der üblichen Form, wie sie auch bei ganzzahligen und reellen Variablen verwendet wird, kann einer Booleschen Größe einer der Werte 'TRUE' und 'FALSE' zugeordnet werden.

Im Programm geschieht das in Zeile 6. Boolesche Größen kann man logisch miteinander verknüpfen, außerdem kann man sie negieren. Der negierte oder komplementäre Wert einer Booleschen Größe wird durch Vorsetzen von 'NOT' bezeichnet. Ist also B wahr, so ist 'NOT' B falsch, ist dagegen B falsch, so ist 'NOT' B wahr. Boolesche Größen sind nützlich zur Darstellung von Vorgängen, die im Wechsel von nur zwei Zuständen bestehen, beispielsweise werden wir sie zur Beschreibung des Hin- und Herfahrens des Zeichengriffels beim Ausfüllen der Rechteckfläche verwenden. Besonders wichtig ist, daß Boolesche Größen an allen Stellen in einem Programm stehen können, an denen auch Aussagen erscheinen dürfen. Das ist im Grunde selbstverständlich, denn Aussagen und Boolesche Werte haben gemeinsam, daß sie mit dem Wahren oder dem Falschen übereinstimmen. Daraus folgt, daß Boolesche Größen zum Aufbau von Weichen dienen können (siehe Abschnitt 2.1.4), dies wird im Programm in den Zeilen 11 und 20 verwendet.

Nun betrachten wir das Programm im einzelnen. In Zeile 5 wird ein Zehntel der Längeneinheit von der Strichstärke des Griffels abgezogen und der so gewonnene Wert der reellen Größe A zugeordnet. Verschiebt man also eine vom Griffel gezogene gerade Linie um A Längeneinheiten parallel zu sich selbst, so erreicht man, daß die beiden Linien sich um ein Zehntel der Längeneinheit überdecken. Auf diese Weise erreicht man die lückenlose Füllung der erwünschten Rechteckfläche. B wird nun auf 'TRUE' gesetzt. In Zeile 7 wird gefragt, ob das zu zeichnende Rechteck Quer- oder Hochformat besitzt. Im Fall des Hochformats werden die Strecken, die zur Füllung der Rechteckfläche dienen, von Programmzeile 17 ab vertikal gezogen. Querformate werden ab Zeile 8 horizontal gefüllt. Durch diese Unterscheidung erreicht man,

daß die füllenden Strecken maximale Länge besitzen, der Zeichenautomat also möglichst wenig durch Richtungsumschaltungen beansprucht wird.

In Zeile 8 taucht eine horizontglobale Standardprozedur auf, die ganz ähnlichen Charakter hat wie die Prozeduren SIN und COS, die wir schon früher verwendet haben. SIGN(V) liefert das Vorzeichen der Zahl V in der Weise, daß SIGN(V) gleich +1 bzw. −1 bzw. 0 ist, je nachdem V größer bzw. kleiner bzw. gleich null ist. Angewandt in Programmzeile 8 bedeutet das, daß H bis auf das Vorzeichen von G−F mit A übereinstimmt. H wird nun im weiteren Fortgang des Programms dazu verwendet, die rechteckfüllende Strecke immer um H Längeneinheiten zu sich selbst parallel zu verschieben. Aufgrund von Zeile 8 wird H dann negativ, wenn beim Rechteck im Querformat die Ordinate F größer ist als die Ordinate G (wir haben ja nicht vorausgesetzt, daß beim Aufruf von REC der Aktualwert von F immer kleiner als oder gleich dem Aktualwert von G ist).

Die momentane Höhe (Ordinate) der füllenden Strecke nennen wir L (im Anklang an „level"). Am Anfang des Zeichenvorgangs geben wir L den Wert der Anfangsordinate F (siehe Zeile 8). Zeile 9 zeichnet den Rahmen des Rechtecks. Nun wird durch die Zeilen 11 bis 14 die füllende Linie wie der Faden im Webstuhl immer hin- und hergezogen. Dabei dient eine Hilfsvariable Z als eine Art Weberschiffchen. Z hat nämlich einmal den Wert von D (Zeile 12), das nächste Mal wieder den von E (Zeile 13). Der Wert der Booleschen Variablen B bestimmt mit Hilfe der Weiche in Zeile 11 abwechselnd die zwei Stellen, an denen Z sich jeweils befinden kann. Durch Zeile 14 wird die füllende Strecke von links nach rechts oder von rechts nach links gezeichnet und anschließend wird B

umgeschaltet. Schließlich fragt Zeile 15, ob die Differenz zwischen dem momentanen *Level* und der Grenzordinate G dem Betrag nach schon kleiner als der Zuwachswert A ist. Ist dies der Fall, so endet das Programm bei der Marke EXIT in Zeile 26.

Da die Zeilen 17 bis 25 homolog zu den Zeilen 8 bis 16 sind, brauchen wir sie nicht zu besprechen. Wir sind damit im Besitz einer Prozedur, die beim Aufruf REC (D,E,F,G,S) eine Rechteckfläche mit den angegebenen Eckpunktkoordinaten zeichnet, wobei wir auf die gegenseitige Lage dieser Koordinaten nicht zu achten brauchen. Die zuletzt erwähnte Eigenschaft hat den Vorteil, daß man D, E, F und G auf einfache Weise durch Zufallsgeneratoren bestimmen kann.

Eine Anwendung von REC zeigt Bild 14. Fast alle in diesem Bild auftauchenden Rechtecke haben Querformat. Das Zeichenprogramm zu Bild 14 ist recht einfach gebaut:

(3)

```
 1 'BEGIN'
 2 'COMMENT' DEMONSTRATION
   RECHTECKFLÄCHE.,
 3 'REAL' P, P1.,
 4 OPEN(0,0)., JI1.=JS1., JI2.=JS2.,
 5 JA1.=0., JE1.=6., JA2.=−80., JE2.=80.,
 6 P.=−130.,
 7 ANFANG..
 8 'IF' P 'GREATER' 130.0 'THEN' 'GOTO' FIN.,
 9 P1.=P+J1., REC(P,P1,0.0,J2,0.4)., P.=P1.,
10 'GOTO' ANFANG.,
11 FIN.. CLOSE
12 'END' DEMONSTRATION
   RECHTECKFLÄCHE.,
```

Man erkennt, daß die zufällig erzeugten Rechtecke von der Abszisse −130 ausgehend (siehe Zeile 6) immer mehr nach rechts geschoben werden. Die Breite des einzelnen Rechtecks wird vom Zufallsgenerator J1, seine Höhe vom Zufallsgenerator J2 bestimmt. Überschreitet der *Abszissenpegel* P den Wert +130, so schaltet das Programm mit Hilfe der Weiche in Zeile 8 ab.

Von kleineren Zusätzen abgesehen, wird die Sprache G3, die ja die früher besprochenen Sprachen G2 und G1 umfaßt, zum Studium der generativen Prozesse ausreichen, denen wir uns widmen wollen. Wir beenden deshalb an dieser Stelle das Kapitel 2, das der Bereitstellung von Werkzeugen gedient hat.

3. Programm und Graphik

Wir sind jetzt im Besitz einer Ausdrucksweise, die den schrittweisen Aufbau einer besonderen Klasse von Texten, nämlich von Programmen, ermöglicht. Die Programme, denen wir uns hier widmen, generieren graphische Information, sobald sie von einer geeigneten technischen Ausrüstung aktualisiert werden. Im Besitz der erwähnten Ausrüstung sind wir in der Lage, allein durch die Abfassung von Texten auch in den Besitz interessanter ästhetischer Information zu gelangen. Bestimmte Texte erzeugen dabei bestimmte Graphiken, wobei die Bestimmtheit dieser Graphiken jedoch in interessanter Weise eingeschränkt ist durch Zufallsprozesse, deren Auswirkung wir absichtlich zulassen. Das Gesamtsystem aus Programmiersprache und technischer Ausrüstung zu ihrer Verarbeitung gleicht einem, nein: ist ein Laboratorium, in dem uns das Experimentieren mit umfangreichem und kompliziertem ästhetischem Material, das wir uns selbst herstellen, immer tieferes Eindringen in die Mannigfaltigkeit ästhetischer Komplexität möglich macht. Was uns auf diesem Weg ästhetischer Forschung noch an Überraschungen — an Bemerkenswertem, Beglückendem, Beunruhigendem, anthropologisch Relevantem — begegnen mag, ist nicht vorauszusehen. Die Programme im vorliegenden Bericht sind ja alle verhältnismäßig einfach gebaut, ihre Länge übersteigt niemals zwei Seiten. Gemessen an den großen, oft den Umfang von fünfzig Textseiten übertreffenden Programmen der industriellen Technik ist das noch sehr wenig. Gar nicht berühren werden wir das Gebiet der zeitlichen Aneinanderreihung von Computergraphiken: Das Problem des ästhetischen Computerfilms.

Unsere experimentelle Arbeit dient einem Ziel: Der gedanklichen Durchdringung ästhetischer Information durch adäquate Begriffe. Diese Begriffe sind Computerprogramme!

3.1. Ästhetische Kategorien

Abschnitt 2.1 schloß mit der Feststellung, daß jeweils eine Graphik und eine Klasse von Automaten einander umkehrbar eindeutig zugeordnet sind. Dabei kann die Graphik von jedem Automaten aus der ihr zugeordneten Klasse erzeugt werden. Die erzeugenden Automaten einer Graphik nannten wir ihre Generatoren. Einen Generator betrachten wir als nicht wesentlich verschieden von dem in der Programmiersprache G geschriebenen Programm, das sein Verhalten festlegt.

Jeder Generator ist eine informationsverarbeitende Maschine insofern, als das Programm Information ist, dem er folgt. Der Generator ist aber ebenso eine informationserzeugende Maschine: Die Information, die er hervorbringt, ist die Graphik, die er generiert. Information stellt sich immer durch Zeichen dar, kompliziert gebaute Information durch Zeichenkonstellationen. Alle Graphiken, die wir bis jetzt programmiert haben, sind in diesem Sinn sicher Zeichenkonstellationen. Wir verknüpfen hier den Begriff des Generators mit dem der Zeichenkonstellation, wie er in AE, Seite 209, eingeführt wird.

Fragt man nach den Struktureigenschaften von Generatoren, so wird man wiederum auf die Programme zurückgeführt, durch die sie realisiert werden. Dies einfach deshalb, weil die relevanten Strukturmerkmale, die nicht von der besonderen technischen Verwirklichung des Ge-

nerators abhängen, nur sein Verhalten betreffen können; das Verhalten folgt aber eindeutig aus der Struktur des Programms.

Als besonders wichtige Strukturelemente haben wir schon in Kapitel 2 die Prozeduren kennengelernt. Die Bedeutung der Prozeduren für die Genese von Graphiken gründet in ihrer Fähigkeit, Elemente oder Bausteine von Graphiken darstellen zu können. Dabei entsprechen einander, wie wir schon in Abschnitt 2.1.6 feststellten, die funktionelle Autarkie der Prozedur einerseits, des Elements der Graphik andererseits. Wir erinnern uns an die Prozedur

(1) FALTER(A,B,FAKTOR)

die durch die Zeilen 2 bis 12 des Programms (1/2.1.6) vereinbart wurde und die es ermöglichte, einzelne Falterfiguren als Bausteine zur Genese sowohl der pfeilartigen Faltersäule in Bild 1, als auch des Falterschwarms in Bild 4 zu verwenden. Einmal am Anfang eines Programms vereinbart, d. h. in ihrem vollen Wortlaut aufgeschrieben, kann eine Prozedur oftmals in einem Programm aufgerufen und dadurch aktualisiert werden. In Programm (1/2.1.6) bauen die Zeilen 14 bis 15 durch wiederholten Aufruf der Prozedur FALTER den Falterpfeil in Bild 1 auf. Das Programm ist eine rahmenartige Maschinerie, die zum Aufruf der Prozeduren dient, die am Anfang des Programms vereinbart worden sind.

Bereits in Abschnitt 2.1.6 erwähnten wir, daß Prozeduren ihrerseits als Rahmen weiterer Prozeduren dienen können und daß solche Prozedurhierarchien hierarchisch gebaute Graphiken generieren. Aus der folgenden Aufgabenstellung ergibt sich ein Beispiel für eine zweistufi-

ge Prozedurhierarchie: Ein zufälliger Schwarm von untereinander ähnlichen Einzelfiguren ist so zu erzeugen, daß ein ausgezeichneter Punkt in jeder der Einzelfiguren in ein Rahmenrechteck fällt und die Größenfaktoren für die Einzelfiguren ebenfalls einem festen Zahlenintervall entstammen. Sind D und E die Abszissen, F und G die Ordinaten der Eckpunkte des Rahmenrechtecks, bestimmen U und V die Intervallgrenzen für den Größenfaktor und ist N die Anzahl der Einzelfiguren, so kann man als wesentlichen Baustein für den Generator des Schwarms eine Prozedur mit dem allgemeinen Aufruf

(2) SCHWARM(D,E,F,G,U,V,N,P)

betrachten. Zur begrifflichen Durchdringung des Prozesses der Genese des Schwarms ist es nicht notwendig, den Prozedurrumpf zu (2), d. h. das Programmstück, das durch (2) aktualisiert werden soll, jetzt schon im einzelnen auszuarbeiten. Die Prozedur SCHWARM besitzt acht Parameter, deren Bedeutung wir — mit Ausnahme der Bedeutung von P — schon kennen. Es kennzeichnet P eine weitere Prozedur, die zur Genese der Einzelfigur im Schwarm dient. Wir setzen dabei voraus, daß der Aufruf von P immer die Form

(3) P(A,B,FAKTOR)

hat, wobei die Koordinaten X=A und Y=B die Position der Einzelfigur festlegen und FAKTOR deren Größe bestimmt. Wie (1) zeigt, wird die Prozedur FALTER ebenso aufgerufen, wie (3), so daß man also durch einen Aufruf der Art

(4) SCHWARM(D,E,F,G,U,V,N,FALTER)

den gewünschten Falterschwarm bekommt. Wird P vereinbart, wie in (6), so erhält man durch einen Aufruf der Art

(5) SCHWARM(D,E,F,G,U,V,N,STAR)

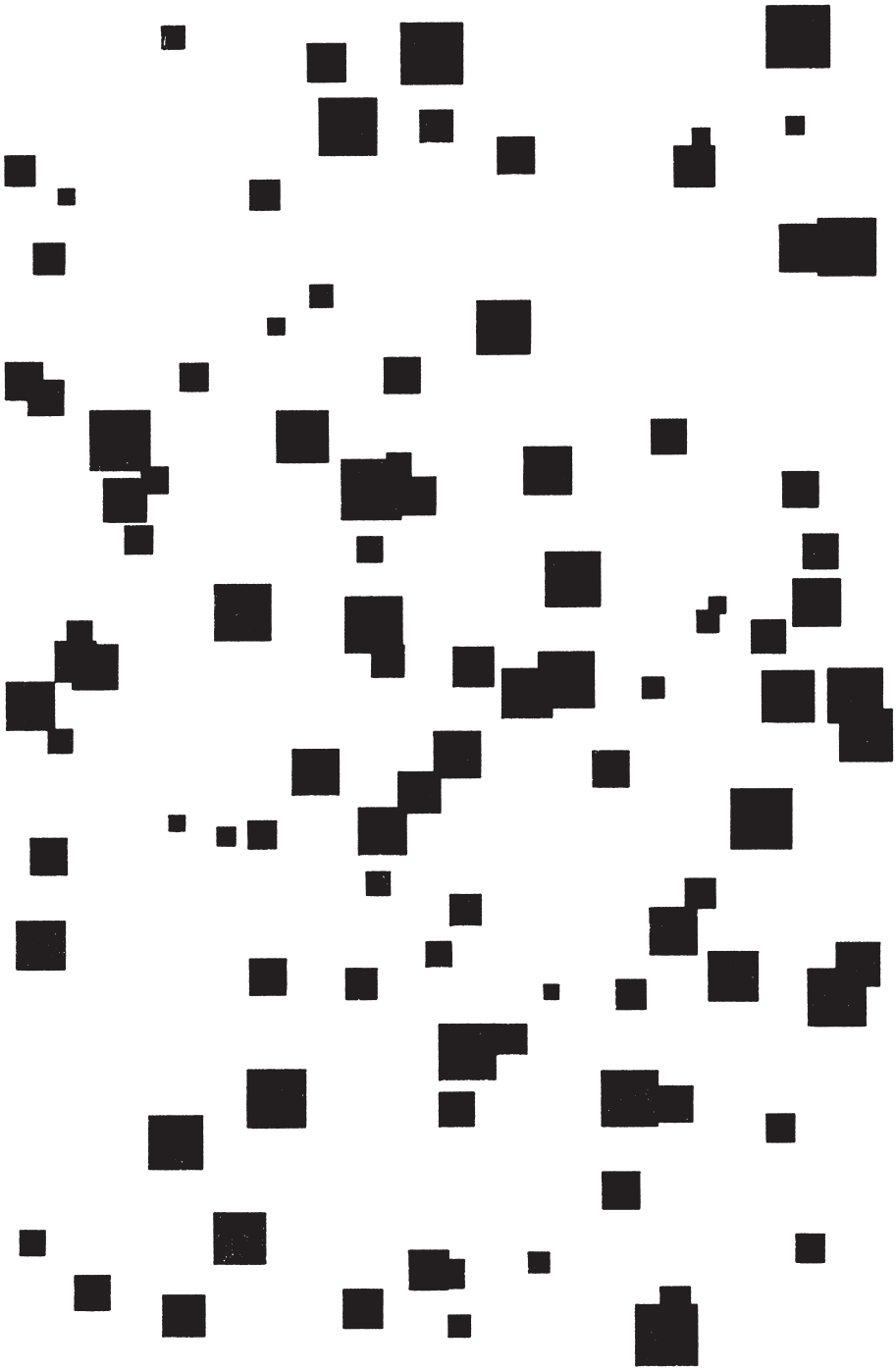
den „Sternhaufen" von Bild 15.

(6)
```
'PROCEDURE' STAR(A,B,FAKTOR).,
'VALUE' A, B, FAKTOR.,
'REAL' A, B, FAKTOR.,
 REC(A−FAKTOR,A+FAKTOR,
 B−FAKTOR,B+FAKTOR,S).,
```

Da der Rumpf der Prozedur STAR nur aus einer einzigen Anweisung besteht, braucht er nicht in 'BEGIN' und 'END' eingeschlossen zu werden. Der Rumpf von STAR verursacht einen Aufruf der Prozedur REC, die wir aus Abschnitt 2.3.2 kennen. Man sieht, daß REC in (6) eine Quadratfläche zeichnet, die mit der Seitenlänge 2 mal FAKTOR symmetrisch um den Punkt X=A, Y=B angeordnet ist. Den Parameter S bestimmt die Strichstärke des Zeichengriffels, der das Rechteck REC flächig ausfüllt, S muß deshalb vor dem Aufruf der Prozedur STAR an einer früheren Stelle im Programm einen Wert zugewiesen bekommen (S ist in bezug auf (6) horizontglobal, wie wir uns in Abschnitt 2.2.1 ausdrückten).

Wir sind sehr abstrakt vorgegangen und haben bis jetzt nur festgestellt, daß Bild 15 durch einen Aufruf der allgemeinen Form (5) hervorgebracht werden kann. Tatsächlich muß man den Parametern D bis N in (5) ganz bestimmte Werte zuweisen, um Bild 15 als individuelle Graphik hervorzubringen. Wie wir wissen, geht diese Wertzuweisung im Rahmen eines Programms vor sich.

Bild 15



Betrachten wir die Prozedur STAR, die wir ja schon durch (6) eingeführt haben, als horizontglobal, so repräsentiert die folgende Anweisung ein Programm und einen mit diesem Programm äquivalenten Generator für Bild 15:

(7) SCHWARM(−120.0,120.0,−80.0,60.0,1.0,
5.0,100,STAR)

Ein Rahmenprogramm, das keine andere Aufgabe hat, als den einmaligen Aufruf einer Prozedur zu organisieren, steht in einer ganz bestimmten Beziehung zu dieser Prozedur: Es besorgt nichts weiter, als eine individuelle Aktivität der Prozedur für individuelle Aktualwerte ihrer Formalparameter.

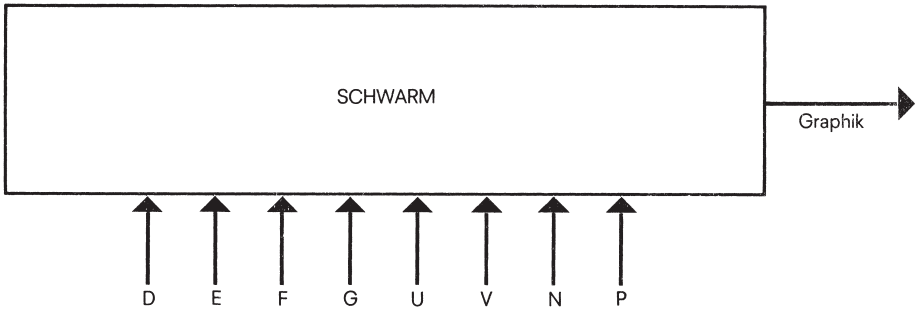
Die Zweistufigkeit der Prozedur SCHWARM ist selbstverständlich ein Spezialfall höherstufiger Verallgemeinerungen. Für uns ist jedoch im Augenblick nur folgendes wichtig: Jede Prozedur kann durch spezielle Besetzung ihrer Formalparameter im Rahmen eines Programmblocks oder durch eine einzige Aufrufanweisung in einen Bildgenerator verwandelt werden. Es ist nun von größter Bedeutung, daß zur Relation Programm — Prozedur eine homologe Relation existiert, die von dem Relationsglied Generator ausgeht. Das zweite Glied in dieser Relation ist der Begriff des Transduktors, der in der Informationstheorie die Wandlung von Zeichenströmen beschreibt (W. R. Ashby: An Introduction to Cybernetics, London 1957).

Man hat also die zwei Beziehungen
Programm — Prozedur
Generator — Transduktor

Transduktoren sind rein gedanklich hergeleitete oder auch technisch realisierte Gebilde, die Eingangsinformation in Ausgangsinformation umwandeln. Sie sind Informationswandler, Überträger von Information von einem Modus in einen anderen. Die Beziehung von Transduktor und Prozedur stellt sich nun ganz einfach auf die Weise her, daß man die Parameter der Prozedur als Eingangsinformation deutet, dementsprechend ist das durch die Prozedur bewerkstelligte Resultat Ausgangsinformation. Beispielsweise kann die Prozedur SCHWARM als Transduktor mit der Eingangsinformation D bis P und mit einem bestimmten Typus von graphischer Ausgangsinformation aufgefaßt werden. Gleichfalls Transduktoren sind FALTER und STAR. Transduktoren können Transduktoren als Teilinstanzen enthalten (man betrachte den Ausdruck (5)). Jeder Transduktor wird durch spezielle Wahl seiner Eingangsinformation in einen Generator verwandelt. Wir werden künftig auch Programmstücke, in denen gewisse Größen als Eingangsparameter deutbar sind, als Transduktoren auffassen. Ferner werden wir die Begriffe Transduktor und Generator nicht immer scharf voneinander unterscheiden: Generatoren betrachten wir als Transduktoren, denen bereits eine bestimmte Eingangsinformation zugewiesen wurde; Transduktoren betrachten wir immer als potentielle Generatoren. Insbesondere aber werden wir jede Prozedur als äquivalent mit dem ihr begrifflich eindeutig zugeordneten Transduktor behandeln.

Die Gewohnheit, Transduktoren auch als *black boxes* zu bezeichnen (siehe Ashby 1957), führt zu einer schematischen Darstellung. Figur 3 zeigt den Transduktor SCHWARM in Gestalt einer black box. Die Parameter der Prozedur werden zu Informationseingängen in den

Transduktor bzw. die black box. Aus naheliegenden Gründen bezeichnen wir die einzelnen Informationseingänge von Transduktoren ebenfalls als Parameter.



Figur 3

Das Prinzip des Transduktors läßt uns zu einer vertieften Auffassung desjenigen Prozesses gelangen, in dessen Verlauf Zeichenkonstellationen entstehen. In AE, Seite 209, heißt es: „Vom Standpunkt der Komposition aus bedeutet die Konstellation eine Verteilung, vom Standpunkt der vorangehenden Konzeption aber eine Auswahl." Zwei dynamische Aspekte des Zeichenprozesses kann man also unterscheiden: Verteilung und Auswahl. Es wird eine der Hauptaufgaben des vorliegenden Abschnitts 3.1 sein, zu zeigen, daß sich die Parameter von Transduktoren unter bestimmten Voraussetzungen als kompositions- und selektionsbestimmende klassifizieren lassen, ja daß der Transduktor ein inneres Gefüge wiederum dynamischer Bedeutung erkennen lassen kann, das ihn als Zusammenschaltung von Kompositions- und Selektor-Transduktoren erweist.

Bevor wir diese Einführung in den Abschnitt 3.1 abschließen, müssen wir noch etwas näher an die Thematik

der „Aesthetica" heranrücken (Bense 1965), deren Begrifflichkeit uns mehr und mehr beschäftigen wird.

Die Zeichenkonstellation, die das Ziel des Zeichenprozesses ist, weist drei Merkmale von so fundamentaler Natur auf, daß wir ihnen den Rang von Kategorien zuerkennen: Innovation, Dispersion und Redundanz. Bezeichnen wir im Blickwinkel der Ästhetik die Ausgangsinformation von Bildgeneratoren als ästhetische, so ist die Innovation der eigentliche Gehalt der ästhetischen Information. Von ihr wird in AE, Seite 225, gesagt: „Die Programmierung rückt gerade das Unvermeidbare ins Bewußtsein, den approximativen, stochastischen Charakter dessen, was hier ästhetische Information, Originalität, Innovation, Schöpfung heißt." Aus diesem Satz folgt auf jeden Fall, daß Innovation eine Art Neuigkeit ist, die der Empfänger ästhetischer Information erfährt.

Redundanz ist der kategoriale Gegenspieler der Innovation. Sie schwächt deren ungemilderte Singularität ab durch Wiederholung: „Redundanz repräsentiert den Betrag, mit dem ein Reglement ästhetisch wirksam wird" (AE, Seite 216). Oder: „Gerade im ästhetischen Zeichenprozeß wird der Zeichenfluß, der eine bemerkenswerte und wesentliche Tendenz nicht auf wahrscheinliche, sondern auf unwahrscheinliche Verteilung der Zeichen hat, erst dann zu einem Informationsfluß, wenn er durch Redundanz, durch Reglement gesteuert ist" (AE, Seite 216).

Zum Begriff der Dispersion heißt es in AE (Seite 276): „Man kann sich vorstellen, daß ästhetische Information bzw. Realisation nicht durch die Redundanzen hervorgerufen wird, die einen ästhetischen Prozeß beeinflussen, also nicht durch den Ballast dieser oder jener Art, der

ihm auferlegt wird, sondern gerade durch entgegengesetzte Einwirkungen. Dokumentarische Information kann dadurch, daß man gewissen Ballast wegnimmt, daß man sie dispersiert, ihren Gehalt an Gegenständlichkeit zerstreut, vermindert, zu einer ästhetischen werden. Dadurch wird nahegelegt, eine negative Redundanz, die *Dispersion,* als wichtiges Moment ästhetischer Prozesse einzuführen. Man kann sich die impressionistische Technik aus der naturalistischen als Folge der Zulassung und Übertreibung der Dispersion im ästhetischen Realisationsprozeß hervorgehend denken." Wir werden sehen, daß in der generativen Graphik vornehmlich die Zufallsgeneratoren Dispersion bewirken, und zwar sowohl im selektiven, als auch im kompositiven Sinn. Später werden wir den Begriff der Dispersion mit dem der Mikroinnovation verknüpfen und dadurch eine fundamentale Beziehung zwischen Innovation und Dispersion aufzeigen.

Nachdem uns jetzt die Grundbegriffe *Generator, Zeichenkonstellation, Transduktor, Innovation, Dispersion* und *Redundanz* geläufig sind, wir außerdem den Programmierapparat der Sprache G zur Verfügung haben, können wir an die Formulierung eines der Grundprobleme der generativen Graphik gehen. Dieses Problem fragt nach den Anteilen von Innovation und Redundanz an der Zeichenkonstellation, die durch einen bestimmten Generator erzeugt wird. Wir erwarten nicht so sehr absolute quantitative Angaben als Antwort auf die gestellte Frage, sondern Auskünfte über den Zusammenhang zwischen der Struktur des Generators und der durch Innovation und Redundanz bestimmten Struktur der generierten Graphik. Da wir unter der Struktur des Generators selbst nichts anderes verstehen, als diejenigen seiner Gefügeeigenschaften, die sein Gesamtverhal-

ten, d. h. seine Funktionalität bestimmen, kann das Problem auf die kurze Formel gebracht werden: *Wie bedingt die Struktur des Generators die Struktur seines Produkts.*

Die nachfolgenden Unterabschnitte werden das gestellte Problem von verschiedenen Seiten beleuchten und schrittweise wenigstens einer Teillösung zuzuführen versuchen.

3.1.1. Selektoren und Kompositoren

Wir betrachten die Prozedur

(1) SCHWARM(D,E,F,G,U,V,N,P),

die schon weiter oben eingeführt und durch Figur 3 schematisch als Transduktor dargestellt wurde.

Welche unter den acht Eingängen führen dem Transduktor SCHWARM eindeutig selektorische und welche führen ihm kompositorische Information zu? Dabei verstehen wir unter selektorischer Information solche, die auf das Ergebnis des Selektionsprozesses unmittelbar Einfluß nimmt; Entsprechendes gelte für den Kompositionsprozeß. Wir werden feststellen, daß die beiden Prozesse sich nicht immer voneinander trennen lassen und daß vor allem der Selektionsprozeß kompositorisch wirksam wird. Man könnte nun vermuten, daß vor allen anderen Parametern die Prozedur P selektorischer Natur ist. Die Begründung könnte folgendermaßen lauten: In einem ersten Schritt ruft man die Prozedur SCHWARM unter Ersetzung von P durch beispielsweise die Aktualprozedur STAR auf. Generiert wird der Sternhaufen. Bei einem zweiten Aufruf läßt man alle Parame-

ter unverändert, mit Ausnahme von STAR, das man gegen eine Prozedur TETRA austauscht, die ein Tetraeder bestimmter Größe zeichnet. Dann ändert sich an der vom Transduktor hervorgebrachten Gesamtkomposition nicht mehr, als daß gewissermaßen die Atome der Komposition gegeneinander ausgetauscht werden (siehe Bild 16). Alle Quadrate werden dabei durch Tetraeder ersetzt. Die Auswahl von STAR oder TETRA aus einem unendlich großen Vorrat möglicher Atomfiguren ist aber ein Vorselektionsprozeß, auf den die Struktur des Transduktors keinerlei Einfluß hat. Bei jener Vorselektion geht gerade das vor sich, was Max Bense als ästhetische Vorentscheidung bezeichnet und was er bei der *ungewohnten Arbeitsteilung im ästhetischen Prozeß* (AE, Seite 338) zur Seite des Programmierers oder Schöpfers des Transduktors rechnet. Der Parameter P beeinflußt jedoch aus einem ganz anderen Grund, zusammen mit den Zahlenparametern U und V, den selektorischen Anteil am Prozeß der Erzeugung des Figurenschwarms. Um dies zu beweisen, müssen wir etwas näher auf den Begriff des Alphabets eingehen.

Alphabete sind Klassen von Grundzeichen. Aus den Grundzeichen eines Alphabets können Texte oder, im Fall graphischer Grundzeichen, Kompositionen zusammengestellt werden. Zur sukzessiven Auswahl der Grundzeichen aus dem Alphabet dient ein Selektionsprozeß. Alphabete können endliche Mengen sein, wie das ABC, das man in der Schule lernt. Es gibt jedoch auch Alphabete, deren Elemente ein Zeichenkontinuum aufbauen. Beispielsweise bilden die Töne, die eine Geige hervorbringen kann, ein solches Alphabet. Im Bereich der Graphik sind die Zeichen überhaupt als ein durchgehendes Kontinuum vorstellbar, aus dem sich Teilkontinua herauslösen lassen. So gehören alle Strecken einer

Bild 16

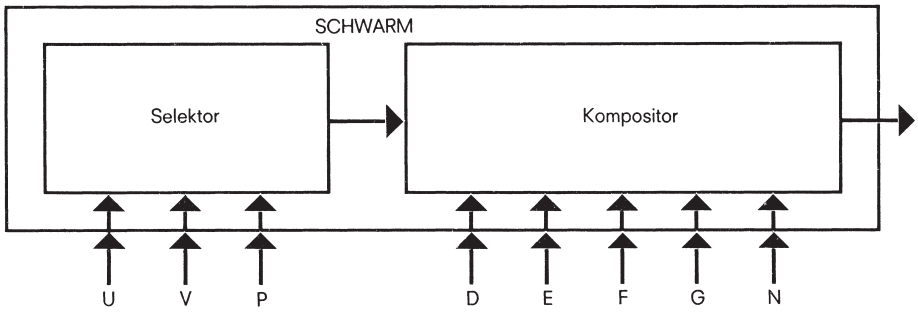
bestimmten Länge einem kontinuierlichen graphischen Alphabet an. Ein einzelnes Zeichen dieses Alphabets ist selektierbar durch Angabe einer reellen Zahl.

Als graphisches Alphabet betrachten wir auch eine Klasse untereinander kongruenter Figuren, deren Größenbereich durch Angabe eines Zahlenintervalls festgelegt wird. Somit wird klar, daß drei feste Werte der oben eingeführten Parameter U, V und P ein Alphabet definieren. Ist z. B. U = 2, V = 3, P = FALTER, so bestimmen diese drei Angaben das Alphabet der Falterfiguren mit einer Größe zwischen 2 und 3. Ob die Größe des Falters durch die Flügelspannweite oder etwa durch die Flügelfläche gegeben wird, hängt vom Bau der Prozedur FALTER ab, eine Indefinitheit des durch die Werte 2, 3 und FALTER bestimmten Alphabets entsteht dadurch nicht.

Einen Selektionsapparat, mit dessen Hilfe ein Einzelzeichen aus dem formelhaft (U,V,FALTER) geschriebenen Alphabet ausgewählt werden kann, kennen wir bereits: Es ist der Zufallsgenerator, dessen Streuintervall durch die Werte von U und V festgelegt ist. Damit können wir die Behauptung, daß U, V und P selektorisch am Prozeß der Erzeugung eines Figurenschwarms, speziell des Falterschwarms, beteiligt sind, als bewiesen betrachten. Wie steht es nun mit den restlichen Parametern D, E, F, G und N im Ausdruck (1)? Auch hier liegt ein Mißverständnis nahe: Bei der Erzeugung des Falterschwarms oder auch des Sternhaufens wird die Einzelfigur an einen bestimmten Ort gebracht. Man kann diesen Vorgang mit Recht als Ortsselektion bezeichnen. Eine Ortsselektion ist jedoch keine Zeichenselektion, d. h. wir betrachten Orte als reine Möglichkeiten für Plazierung von Zeichen, nicht schon als Zeichen selbst. Sprechen wir ohne kennzeichnenden Zusatz von Selektion, so meinen

wir immer Zeichenselektion. Was andererseits Ortsselektion wirklich hervorbringt, ist eine Zeichenverteilung. Die Zeichenverteilung wiederum legt die extensionale Beschaffenheit der generierten Graphik fest, anders ausgedrückt: Die Komposition. Ohne jeden Zweifel wirken nun die Parameter D, E, F, G zeichenverteilend, denn sie bestimmen den Rechteckrahmen, in dem die durch P gestaltdeterminierten Einzelfiguren untergebracht werden. Unbedingt kompositiv ist N, denn N bestimmt auf keinen Fall eines der Einzelzeichen als solches, aus denen sich die Graphik aufbaut. Der Sternhaufen als Zeichenkonstellation wird jedoch sehr stark von N beeinflußt, was man sich leicht dadurch klarmachen kann, daß man sich die Ergebnisse der Aufrufe der Prozedur SCHWARM für die Werte N = 1 und N = 1000000 vorzustellen versucht.

Wir halten das Ergebnis fest: Die Parameter U, V, P des Transduktors SCHWARM sind selektorisch, die Parameter D, E, F, G, N sind kompositorisch. Dieses Ergebnis können wir schematisch darstellen, wie Figur 4 zeigt.



Figur 4

Der Transduktor erweist sich also als eine Hintereinanderschaltung von zwei Transduktoren, von denen der

eine ein Selektor, der zweite ein Kompositor ist. Nach Absorption eines Tripels U, V, P ist der Selektor in den Stand versetzt, eine beliebig lange Folge von untereinander kongruenten Figuren abzugeben. Diese Folge dient zusammen mit einem Quintupel D, E, F, G, N als Eingangsinformation des Kompositors, der am Ausgang den fertigen Figurenschwarm liefert.

3.1.2. Transduktorstruktur und Programmstruktur

Läßt sich die durch Figur 4 wiedergegebene Struktur des Transduktors SCHWARM im Aufbau des Rumpfes der Prozedur (2/3.1) wiedererkennen? Diese Frage kann nur durch die explizite Programmierung des Rumpfes beantwortet werden. Eine mögliche Fassung der Prozedur ist folgende:

```
(1)   1  'PROCEDURE' SCHWARM(D,E,F,G,U,V,N,P).,
      2  'VALUE' D, E, F, G, U, V, N.,
      3  'INTEGER' N., 'REAL' D, E, F, G, U, V.,
      4  'PROCEDURE' P.,
      5  'BEGIN' 'INTEGER' I.,
      6   JA1.=U., JE1.=V.,
      7   JA2.=D., JE2.=E., JA3.=F., JE3.=G.,
      8  'FOR' I.=1 'STEP' 1 'UNTIL' N 'DO'
      9   P(J2,J3,J1)
     10  'END' SCHWARM.,
```

Dabei ist der Aufruf der Prozedur P nach dem Vorbild der Prozedur FALTER modelliert worden, deren Aufbau wir in Abschnitt 2.1.6 durchgeführt haben. D. h., daß in P(A,B,FAKTOR) die Koordinaten A und B den Ort der Figur und FAKTOR ihre Größe bestimmen.
Die Fassung (1) von SCHWARM zeigt zwar deutlich die

Einbringung der Parameterwerte D bis N in den Zeilen 6 bis 8 und ebenso deutlich die Stellung von P in Zeile 9, jedoch von einer Hintereinanderschaltung von zwei Instanzen, wie in Figur 4, ist zunächst nichts zu bemerken. Dies liegt jedoch daran, daß wir Zeile 9 außerordentlich kompakt programmiert und in ihr die Funktion von vier verschiedenen Prozeduraufrufen zusammengezogen haben. Die Symbole J1 bis J3 rufen ja ihrerseits die Funktionsprozeduren der Zufallsgeneratoren auf (vergleiche (1/2.2.1), Zeilen 6 bis 15). Ändern wir Zeile 9 folgendermaßen ab, wobei wir allerdings noch an Zeile 5 die Vereinbarung der reellen Größen A und B mindestens in Gedanken anzuschließen haben, so wird das Ablaufverhalten des Programms besser erkennbar:

```
(2)   9a  'BEGIN' A.=J2., B.=J3.,
          P(A,B,J1) 'END'
```

Die Zeile 9a birgt allerdings insofern eine Überraschung in sich, als sie einer Umkehrung der Reihenfolge der Transduktoren in Figur 4 entspricht: Zeile 8 ist kompositiv wirksam, A.=J2 und B.=J3 wirken ebenfalls kompositiv, weil ortsselektierend. Erst danach wird innerhalb des Aufrufs von P durch J1 die Größenselektion für die Einzelfigur im Schwarm vorgenommen, danach wird durch Auswertung von P die Einzelfigur als solche ins Leben gerufen. Die Teiltransduktoren in Figur 4 sind also vertauschbar, was der einfachen Tatsache entspricht, daß man die Einzelfigur auch nach der Auswahl ihres Platzes aufzeichnen kann, anstatt vorher. Es ist jedoch interessant, daß die Prozedur SCHWARM langwieriger zu programmieren ist, wenn man sich auf die Forderung versteift, sie möge der Ablauffolge von Figur 4 entsprechen. Dann muß man nämlich die Koordinaten der Einzelfigur erst einmal relativ zum Koordinatenur-

sprung errechnen und sie nach der Ortsselektion verschieben. Dies erfordert umfangreiche Informationszwischenspeicherungen, wenn die Einzelfigur durch viele verschiedene Koordinaten bestimmt ist.

Am Beispiel des Transduktors SCHWARM und der mit ihm äquivalenten Prozedur haben wir dargetan, daß sich die Parameter von Transduktoren nach kompositions- und selektionsbestimmenden sortieren lassen und daß sich daraus eine Zerlegung der Transduktoren in Teiltransduktoren ergibt. Diese Zerlegung spiegelt eine Aufteilung der Verhaltensstruktur der Transduktoren in komponierende und selektierende Aktivitäten wider, die im einfachsten Fall streng zeitlich hintereinander ablaufen. Die Verhaltensstruktur des Transduktors läßt sich bei geeigneter Programmierung auch im Aufbau der mit dem Transduktor äquivalenten Prozedur nachweisen. Die Aufteilung der Verhaltensstruktur des Transduktors ist in einem gewissen Umfang willkürlich: Manchmal ist die Reihenfolge von Aktivitäten umkehrbar.

Schon am Anfang von Abschnitt 3.1.1 sprachen wir jedoch die Vermutung aus, daß der Selektionsprozeß in den Kompositionsvorgang eingreift. Dieser Erscheinung widmen wir den nächsten Abschnitt.

3.1.3. Komponierende Selektion

Ein einfaches Beispiel für die figurdeterminierende Prozedur P ist

```
(1)   'PROCEDURE' KREIS(P,Q,RADIUS).,
      'VALUE' P, Q, RADIUS.,
      'REAL' P, Q, RADIUS.,
      'BEGIN'
```

```
LEER(P+RADIUS,Q).,
ZIRK(P,Q,-RADIUS,P+RADIUS,Q)
'END' KREIS.,
```

Zur Formulierung des Prozedurrumpfes wird die Sprache G3 verwendet (siehe Abschnitt 2.3.1). KREIS(P,Q, RADIUS) zeichnet einen Kreis mit Radius RADIUS um den Mittelpunkt X=P und Y=Q.

wir tätigen nun zwei Aufrufe von SCHWARM unter Verwendung von KREIS:

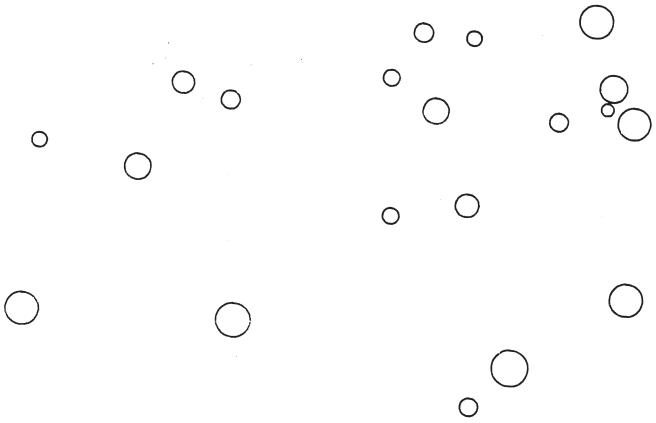
(2) SCHWARM(30.0,100.0,−50.0,50.0,1.0, 3.0,20,KREIS)

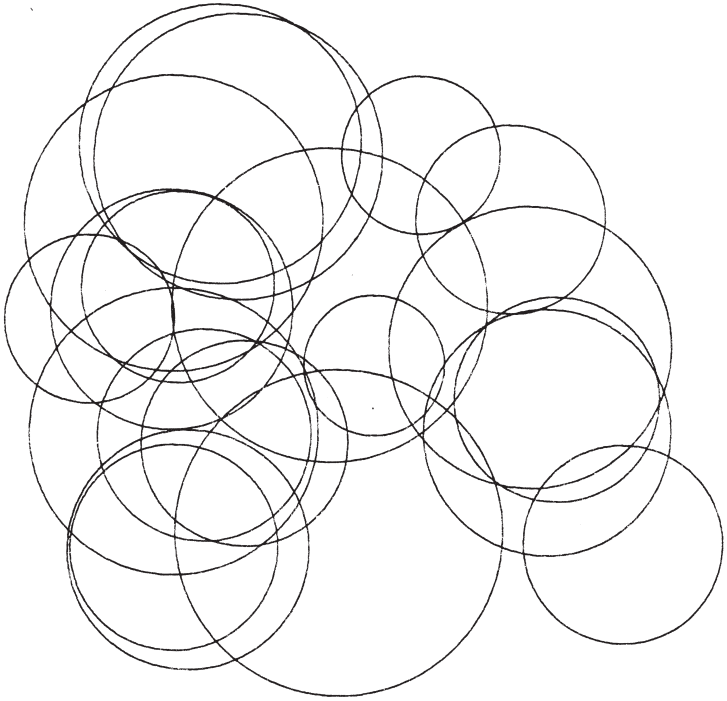
(3) SCHWARM(−100.0,−30.0,−50.0,50.0, 10.0,30.0,20,KREIS)

Bild 17 ist das Ergebnis der Aufrufe (2) und (3). In beiden Teilbildern von Bild 17 sind die Rahmenrechtecke, in denen die Mittelpunkte der jeweils 20 Kreise liegen, genau gleich groß, wie man auch durch Inspektion der ersten vier Parameter beider Prozeduraufrufe (2) und (3) feststellen kann. Der wirksame Unterschied zwischen beiden Aufrufen besteht lediglich darin, daß das Schwankungsintervall für den Kreisradius beim zweiten Aufruf zehnmal so groß ist als beim ersten. Gleichwohl gehören die beiden Teilbilder zwei völlig verschiedenen Typen ästhetischer Information an, das erste könnte man als Haufen bezeichnen, das zweite ist eine echte Gestalt.

Man muß zugestehen, daß das Prinzip der Prozedur SCHWARM, einen Schwarm, d. h. eine Anhäufung voneinander getrennter Einzelfiguren zu zeichnen, durch den Aufruf (3) überdehnt ist. Wie die eines umgangssprachlichen, so wird auch die Bedeutung eines formali-

Bild 17





sierten Begriffes (SCHWARM ist ein solcher) verfälscht, wenn man ihn in Bereichen benutzt, für die er von seinem Entwerfer nicht gedacht war. Soll SCHWARM ein theoretisch ästhetisches Modell für die Genese von schwarmartigen Gebilden sein, so muß die Variationsbreite der Parameter von SCHWARM entsprechend eingeschränkt werden.

Freilich hat das eine Teilbild von Bild 17 seine ästhetische Daseinsberechtigung so gut wie das andere. In der Tat ist das untere (im Hochformat betrachtet) höher organisiert, als das obere. Es eignet ihm innerer Zusammenhang, der durch die Überlappung der zwanzig Einzelfiguren zustande kommt. Die Auswirkung einer solchen Überlappung durch Ausdehnung der Einzelfiguren eines Schwarms bezeichnen wir auch als Überlagerungs- oder Überschiebungseffekt. Der Überschiebungseffekt bringt eine gegenseitige Verflechtung der Einzelfiguren mit sich und wirkt dadurch gestaltschaffend.

Das untere Teilbild als Komposition verdankt seine typische Konstitution dem Betrag des Radius der Einzelkreise. Die Intervallgrenzen für den größenbestimmenden Parameter der Einzelfigur, im vorliegenden Fall ist es der Radius des Einzelkreises, wurden jedoch in Abschnitt 3.1.1 als selektorische Parameter klassifiziert. Ein selektorischer Parameter eines Transduktors kann also in starkem Maße auch die Komposition der Zeichenkonstellation beeinflussen, die der Transduktor hervorbringt. Wie wir gesehen haben, wird diese Erscheinung gegebenenfalls durch Verschiebung der Intervallgrenzen einer einzigen Zahlenvariablen des Transduktors ausgelöst. Verflechten sich durch den Einfluß selektorischer Parameter auf die Komposition Einzelzeichen der Gesamtkonstellation, so tritt eine Bereicherung der Kon-

stellation durch die Einlagerung neuer Beziehungen zwischen ihren Elementen ein. Es erhöht sich der ästhetische Informationsgehalt der Komposition. Die Innovation der Komposition steigt an, wie wir später sagen werden.

Eine merkwürdige Konsequenz aus dem Überlagerungseffekt ist folgende: Tritt er durch geeignete Wahl des Werts eines Eingangsparameters auf, so kann sich das Bild ändern, das man sich von der Innenstruktur des Transduktors zurechtgelegt hat. Eine säuberliche Trennung, wie in Figur 4, von Selektoren und Kompositoren als Teilinstanzen des Transduktors kann unmöglich werden. Im Grunde ist das aber nur eine andere Wendung der Regel, daß Begriffe unbrauchbar werden, wenn man sie an falscher Stelle benutzt. Klar ist jetzt, daß die praktische Aufspaltung ästhetischer Geneseprozesse in selektorische und kompositorische Aktivitäten, durch geeignete Parametereingrenzungen, zwar die theoretische Durchdringung der Prozesse erleichtert, jedoch auch das Auftreten unvorhergesehener bereichernder ästhetischer Informationen unterdrücken kann.

3.1.4. Makroinnovation und Mikroinnovation

Die Begriffe *Generator, Programm, Transduktor, Prozedur,* bezeichnen alle das gleiche, nämlich die Instanz, die ästhetische Information hervorbringt. Jeder Generator ist ein Modell des künstlerischen Schöpfungsprozesses und insofern ist auch sein Produkt das Modell eines Kunstwerks. Es ist, wie Max Bense sagt, *„Künstliche Kunst".* Der Vorteil eines jeden gutgebauten Modells besteht darin, daß es durch seine finite Maschinerie die überaus verwickelten Verhältnisse des Originals überschaubar macht. Freilich reduziert das Modell, d. h. es

streicht Einzelheiten des Originals aus, die man natürlich am liebsten auch abbilden würde, die man aber im weiteren Fortschritt der Forschung durch verfeinerte Modelle zu erfassen hofft. Auch in AE wird die Reduktion als methodischer Vorteil gewertet (Seite 138): „Dieses auf das ästhetisch Wesentliche reduzierte Kunstwerk ist natürlich auch in jedem Fall ein reduziertes Kunstwerk. Das ästhetisch Wesentliche besteht in der Zeichenthematik. Der ästhetische Prozeß ist, wie gesagt, ein Zeichenprozeß. Das reduzierte Kunstwerk läßt deutlicher als das nichtreduzierte den ästhetischen Modus der Zeichenwelt erkennen. Der Nachweis der Zeichen und der Zeichenwelt gehört zu den wichtigsten Ermittlungen der ästhetischen Analyse des Kunstwerks."

Wie die letzten Abschnitte gezeigt haben, macht der Begriff des Generators bzw. des Transduktors besonders gut die Anteile am ästhetischen Zeichenprozeß sichtbar, die in AE als fundamental nachgewiesen werden: Zeichenauswahl und Zeichenverteilung. Im vorliegenden Abschnitt betrachten wir nun eine andere Eigenheit der Generatoren näher, nämlich ihre durch die Funktion von Zufallsgeneratoren induzierte Spontaneität. Der Zeichenautomat (diesen Begriff prägten wir noch vor dem des Generators) erweckt durch die bewußtlose Hurtigkeit, mit der er vor sich hin arbeitet und seine Produkte hervorsprudelt, den Eindruck eines in einem Gehäuse versteckten Somnambulen oder Trancemalers. Unwillkürlich muß man an das denken, was Bense über die Experimente von Micheaux unter dem Einfluß von Mescalin sagt (AE, Seite 168): „Zweifellos vollzog sich in diesen Versuchen das künstlerische *Machen* nicht als ein jeglichen Naturprozeß transzendierender Vorgang des Bewußtseins, und das Resultat konnte unter diesem Aspekt nicht rein *Kunstschönes* sein. Tatsächlich erweist sich

die aufgezeichnete *Zeichenwelt* als eine Welt, die sich durch die Strukturen des wahrscheinlicheren Zustandes gleichmäßiger Verteilung beschreiben läßt, und der ästhetische Prozeß, dem sie ihre Entstehung verdankt, verläuft innerhalb eines veränderten und reduzierten Bewußtseins, dem die operative Einheit durch ein Selbstbewußtsein nicht gegeben ist, durchaus als Naturprozeß, also in Richtung physikalischer Vorgänge, die *Naturschönes* produzieren, aber *Kunstschönes* vortäuschen."

Hier ist mehreres für uns wichtig! Zwar sind die Zufallsgeneratoren, die wir benutzen, Pseudozufallsgeneratoren in dem Sinn, daß ihre Wirkung auf der Ausnützung von zahlentheoretischen Gesetzmäßigkeiten beruht. An ihre Stelle könnten jedoch genausogut Zufallsquellen treten, deren Basis der Zerfall einer radioaktiven Substanz ist. Insoweit, als der Zeichenprozeß von zahlentheoretisch funktionierenden Zufallsgeneratoren gesteuert wird, läuft er deshalb durchaus mit einem physikalischen Vorgang konform. Bemalt nun der Trancemaler ein viereckiges Blatt Papier und läßt er sich so durch den Zwang des Papierrandes in seiner Bewegungsfreiheit einschränken, so verhält er sich nicht viel anders als ein Zeichenautomat, der aufgrund seines Programms innerhalb vorgegebener Feldgrenzen operiert, nicht anders beispielsweise auch als Regentropfen, die zu Beginn des Regens auf den weißen Zementboden eines Hofgevierts fallen. Man vergleiche dazu die Bilder 15 und 16 sowie das Tropfenmuster der einen Hälfte von Bild 17. Auch Bild 4 kann man sich entstanden denken durch möglichst absichtsloses Setzen der Falterfiguren von menschlicher Hand oder auch als eine Momentaufnahme eines anfliegenden Schwarms von Wanderfaltern.
Es ist richtig, daß die Transduktoren, die Bildgeneratoren der generativen Graphik, absichtsvoll programmiert

werden und daß auch ihre Eingangsinformation durch eine Vorentscheidung bestimmt wird. Durch die Verwendung von Zufallsgeneratoren aber gibt man zwar absichtlich, jedoch endgültig Entscheidungsmöglichkeiten aus der Hand und überläßt die Bildentstehung einem blinden Walten.

Kehren wir noch einmal zu Benses Bemerkung über die Zeichenwelt des Mescalinrausches zurück, und zwar zu der Stelle, wo gesagt wird, daß diese Welt sich *durch die Strukturen des wahrscheinlicheren Zustandes gleichmäßiger Verteilung beschreiben läßt!* Der Falterschwarm, die Regentropfenmuster, sind im Automatenmodell, wie in der Natur, hochwahrscheinliche Strukturen. Es kommt nicht vor, daß die Regentropfen das Wort *Gott* auf den Boden malen oder die Falter in einem kubischen Gitter fliegen und auch die zahlentheoretischen Pseudozufallsgeneratoren bringen nichts dergleichen zustande. Dynamisch gesehen ist gleichmäßige Verteilung ein Vorgang, der Vorentscheidungen, als den Quellinformationen irgendwelcher Herkunft, gerade entgegenwirkt. Auch folgendes Beispiel gehört hierher: Seit der Entwicklung der Molekulargenetik wissen wir, daß es texthafte Urinformation ist, die einen Apfelbaum lauter Äpfel hervorbringen läßt, daß aber ein Prozeß ganz anderer Art dem einzelnen Apfel Unrundheit, Färbung und Runzelung der Schale gibt, wie sie ähnlich jeder andere Apfel am Baum auch an sich hat, allerdings keiner in genau dieser Individualität.

Wir verknüpfen diese Überlegungen mit dem Begriff des Alphabets, den wir schon in Abschnitt 3.1.1 verwendet haben. Dort sagten wir, daß der Selektionsvorgang, der eine Teilaktivität des Zeichenprozesses ist, als wiederholte Auswahl von Zeichen aus einem Alphabet aufge-

faßt werden kann. Im Fall des Transduktors SCHWARM (siehe Figur 4) entnimmt der Selektor Einzelzeichen aus einem Alphabet, die alle die von der Prozedur P bestimmte Gestalt haben. Durch die Gestaltuniformität der Einzelzeichen ist dafür gesorgt, daß unter den Einzelzeichen, die der nachgeschaltete Kompositor nun verteilt, kein außergewöhnliches ist, kein *weißer Rabe.* Schon die vom Selektor abgegebenen Einzelzeichen ähneln einander, wie die Äpfel am Baum. Vergegenwärtigen wir uns nun den zeitlichen Ablauf des Geneseprozesses: Nach Eingabe der Eingangsinformation gibt der Transduktor als Ausgangsinformation eine Zeichenkonstellation ab. Dieser Vorgang kann zeitlich wiederholt werden. Sieht man von der Modelleigenschaft der Pseudozufallsgeneratoren im Transduktor ab, nach jedem Informationsausstoß neu initiiert werden zu können, so ruft die in einem geeigneten Zeitrhythmus wiederholte Eingabe der immer gleichen Eingangsinformation eine zeitlich verteilte Folge von Zeichenkonstellationen hervor, die einander ähneln wie ein früchtetragender Apfelbaum einer bestimmten Sorte dem anderen. Noch einmal: Denkt man sich bei diesem Ablauf Pseudozufallsgeneratoren am Werk, so werden diese nicht nach jedem Ausstoß des Transduktors neu geladen, sondern laufen einfach weiter. Mit dieser Vorstellung stimmt überein, daß in der Prozedur SCHWARM am Anfang von Abschnitt 3.1.2 gar keine Initiierung der Zufallsgeneratoren vorgenommen wird. Man kann den Vorgang der periodisch wiederholten Eruption des Transduktors also sehr einfach durch die Anweisung

```
(1)   'FOR' I.=1 'STEP' 1 'UNTIL' K 'DO'
        SCHWARM(D,E,F,G,U,V,N,P)
```

beschreiben, in der D bis P den Charakter von Konstan-

ten haben und K die Anzahl der gewünschten, aufeinanderfolgenden Eruptionen des Transduktors bezeichnet. Zwischen den einzelnen Eruptionen findet keine Wieder-Initiierung der am Prozeß beteiligten Zufallsgeneratoren statt.

Ein konkretes Beispiel zu dem allgemeinen Aufrufschema (1) erhält man, wenn man sich an den speziellen Aufruf (7/3.1) der Prozedur SCHWARM anlehnt, TETRA an die Stelle von STAR setzt, schließlich für K den Wert 2 wählt:

(2) 'FOR' I.=1 'STEP' 1 'UNTIL' 2 'DO'
SCHWARM(−120.0,120.0,−80.0,60.0,1.0,
5.0,100,TETRA)

Der erste Schleifendurchlauf der Laufanweisung (2) liefert Bild 16, der zweite das neue Bild 18. Die Bilder 16 und 18 unterscheiden sich deutlich voneinander und ähneln einander doch nach dem Modus, nach dem auch ein früchtetragender Apfelbaum einem zweiten von der gleichen Sorte ähnelt.

Wir ziehen eine Zwischenbilanz: Ein bestimmter Transduktor, der zu einem Zeitpunkt mit einer ganz bestimmten Eingangsinformation beliefert wurde, gleicht einem einmal vormotivierten Trancemaler. Auch dieser würde eine Serie von Bildern mit typischer, durch einmalige Motivation determinierter Ähnlichkeit hervorbringen können. Allein durch die Kontingenz der Aktivität in Trance, die nur durch die anfängliche Motivation beschränkt wäre, würde die Individualität der Einzelbilder der Serie gegeben und gesichert sein. Die Produkte der automatischen Aktivität zeichnen sich durch Einförmigkeit aus und dies sogar in zwei Ebenen: Sowohl die

Bild 18

Komponenten eines Einzelbildes, als auch die Einzelbilder einer Serie ordnen sich jeweils einem Typus so sehr unter, daß sich keine *Ausreißer* unter ihnen finden. Die Individualität der Einzelbilder einer Eruptionsserie eines bestimmten Transduktors — im Sinn des Äquivalenzprinzips kann auch ein Trancemaler als Transduktor betrachtet werden — ist allein die Wirkung der an der Bildgenese beteiligten Zufallsprozesse.

Man kann eine Bildserie, die von einem Transduktor auf der Basis einer bestimmten Eingangsinformation erzeugt wird, auch als Alphabet auffassen. Die Bilder 16 und 18 stellen ein solches Alphabet dar, ebenso die Bilder 19 und 20, deren Erzeugungsgesetz besonders primitiv ist: Zehn Punkte werden gestreut und dann untereinander verbunden. Gegenüber den Zeichenalphabeten, aus denen selektorische Transduktoren auswählen, sind die Bildserien jedoch Alphabete einer höheren Stufe, Superalphabete sozusagen. Alle Zeichen eines Superalphabets ähneln einander, keines sticht besonders hervor. Jedes Zeichen des Superalphabets jedoch ist ein einzigartig gestaltetes Individuum, nicht zwei Zeichen gleichen einander. Es sei denn, dieser methodologische Vermerk sei gestattet, man gehe so weit in die Zukunft, daß die Pseudozufallsgeneratoren sich zu wiederholen beginnen. Ohne Inkonsistenzen befürchten zu müssen, kann man aber von der Periodizität der verwendeten Zufallsgeneratoren absehen.

Wir sind nun sehr nahe an einer verhältnismäßig reinen Erfassung des Begriffs der Innovation. Was die Zeichen — es sind Zeichenkonstellationen — des Superalphabets jetzt noch unterscheidet, was jeder Zeichenkonstellation ihre Individualität verleiht, hängt in keiner Weise von der Eingangsinformation des Transduktors ab, sondern

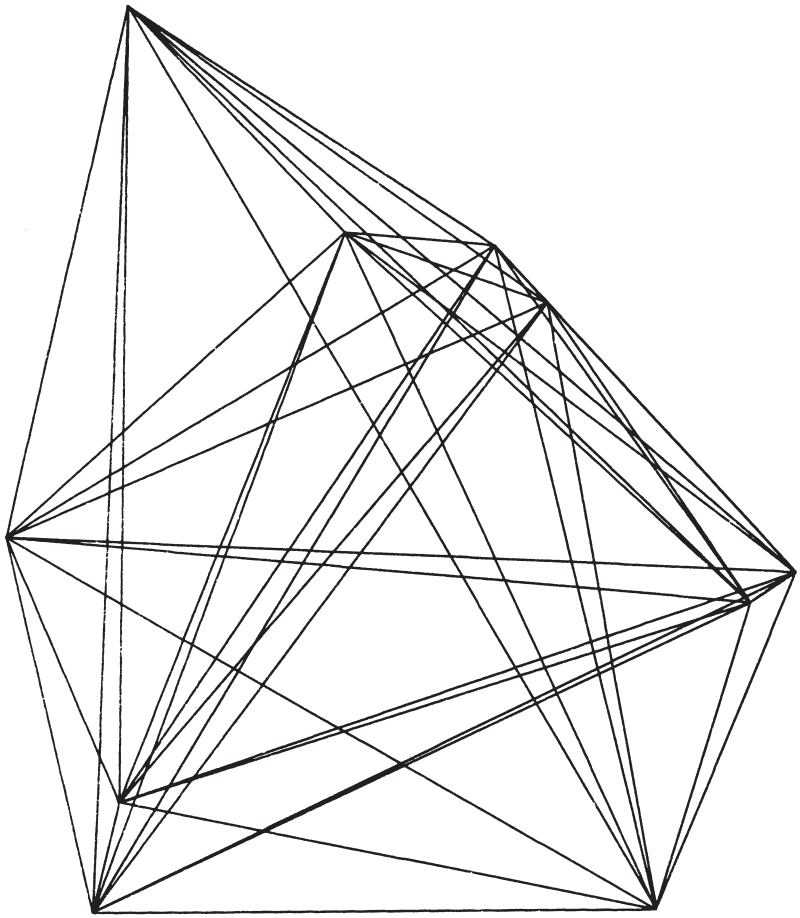
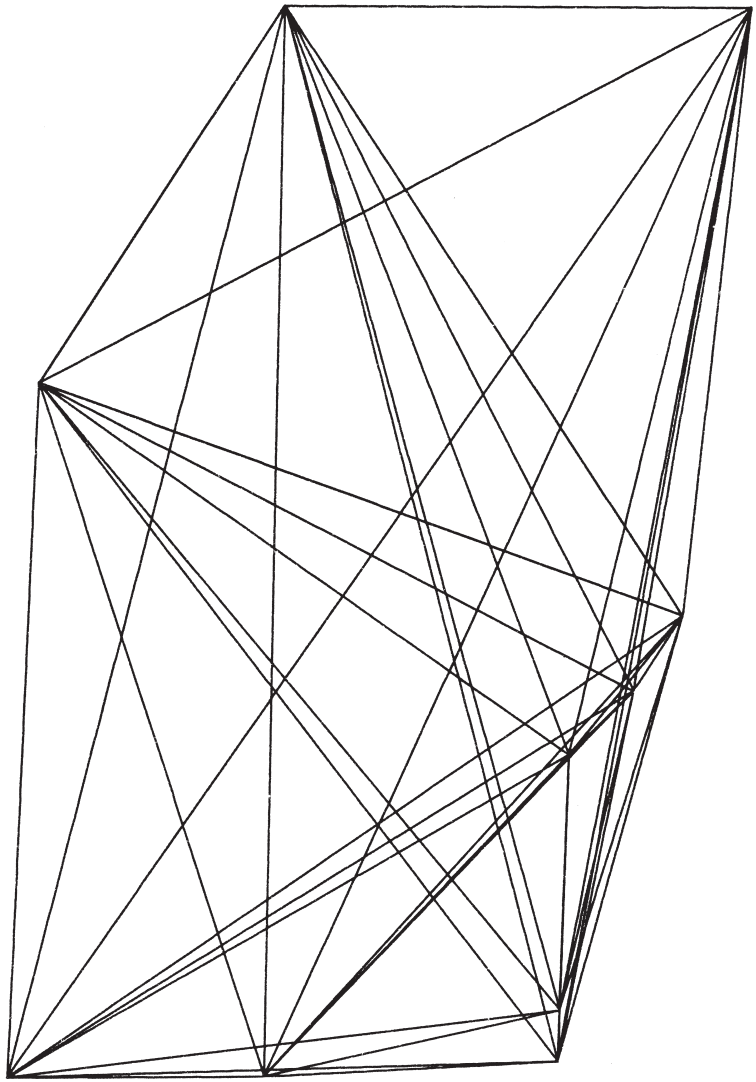


Bild 19

Bild 20



wird allein durch die Funktion der in ihn eingebauten Zufallsgeneratoren bestimmt. Diese aus der Autarkie des Generators geborene, unwiederholte Einzigkeit der Zeichenkonstellation bedeutet in einem ganz natürlichen Sinn Neuartigkeit, sie ist deshalb ein Modus der Innovation. Wir bezeichnen diesen Modus als Mikroinnovation.

Mikroinnovation ist ein Produkt der Funktionalität von Zufallsgeneratoren. Vorhin verglichen wir die von Zufallsgeneratoren gesteuerte Dynamik mit Aktivität in Trance, auch mit Naturprozessen, die, nur durch Zwangsbedingungen eingeengt, absichtslos Ereignisketten hervorbringen. Wir bewegen uns hier in dem Bereich, den Max Benses Mikroästhetik erschlossen hat. Mikroästhetik erfaßt ja gerade diejenigen Anteile am ästhetischen Zeichenprozeß, die nicht beschreibbar sind als absichtsvolle, globale Setzungen uniform bedeutungsgeladener Zeichenkomplexe, sondern sich an der Basis des Zeichenprozesses, in dessen Feinstruktur, in der Region der Elementarzeichen und Elementarereignisse vollziehen. „In erster Näherung verstehen wir nun unter Makroästhetik die Theorie der wahrnehmungsmäßig und vorstellungsmäßig zugänglichen und evidenten Bereiche am ästhetischen Gegenstand bzw. Kunstwerk, während die Mikroästhetik die Theorie der wahrnehmungsmäßig und vorstellungsmäßig nicht direkt zugänglichen und nichtevidenten Bereiche am Kunstwerk bzw. ästhetischen Gegenstand darstellt; sie entwirft das System der ästhetischen Elemente, der Zeichen und ihrer Prozesse" (AE, Seite 142). Wir können dem Begriff der Mikroästhetik einen genauen Sinn dadurch verleihen, daß wir ihn konsequent mit dem Hauptinstrument der generativen Graphik verknüpfen, nämlich dem Generator. Mikroästhetik ist in diesem Sinn die Theorie der Generatoren im

Hinblick auf ihre von den Zufallsgeneratoren abhängigen Eigenschaften.

3.1.5. Redundanz und Makroinnovation

Am Anfang dieses Abschnitts 3.1 sagten wir, daß die zweite fundamentale Kategorie der Ästhetik, nämlich die Redundanz, die ungemilderte Singularität der Innovation durch Wiederholung abschwäche. Da wir jetzt wissen, daß das schlechthin Unwiederholbare an der einzelnen Zeichenkonstellation, die der Generator hervorbringt, durch die Arbeit der Zufallsgeneratoren bewirkt wird, ist klar, daß die Redundanz der Zeichenkonstellationen andere Quellen haben muß. Hält man die Eingangsinformation des Transduktors konstant, läßt man ihn dann eine Serie von Konstellationen erzeugen, so gewinnt man eine Kette von Metamorphosen der Eingangsinformation. Die Invarianz bei der schrittweisen Wandlung wird jedoch außer von der Eingangsinformation durch genau die Teilstrukturen des Transduktors bestimmt, die mit der Funktionalität der in ihn eingebauten Zufallsgenneratoren nichts zu tun haben. Jene von den Zufallsgeneratoren abgetrenntenTeilstrukturen bestimmen das Nicht-Zufällige an der Funktionalität des Transduktors, anders ausgedrückt, bestimmen sie Reglement. Nichts paßt besser hierher als eine Wiederholung eines Zitats aus AE: „Gerade im ästhetischen Zeichenprozeß wird der Zeichenfluß, der eine bemerkenswerte und wesentliche Tendenz nicht auf wahrscheinliche, sondern auf unwahrscheinliche Verteilung der Zeichen hat, erst dann zu einem Informationsfluß, wenn er durch Redundanz, durch Reglement gesteuert ist" (AE, Seite 216).

Für den Programmierer stellt sich die Struktur des Transduktors dar als die Struktur des Programms in der

Sprache G, das die Verhaltensweise des Transduktors beschreibt. Entsinnen wir uns nun des Grundproblems der generativen Graphik, wie wir es am Anfang des vorliegenden Abschnitts 3.1 formulierten: „Wie bedingt die Struktur des Generators die Struktur seines Produkts?", so können wir jetzt eine summarische Lösung angeben: Die mikroinnovativen Anteile am Produkt hängen von der Aktivität der Zufallsgeneratoren des Generators, die redundanten von seiner Weichen- und Schleifenstruktur ab, insoweit diese nichts mit der Maschinerie der Zufallsgeneratoren zu tun hat. Ferner ist die Redundanz abhängig von den Werten gewisser Variablen im Generator, wobei diese Werte die Eingangsinformation des Transduktors bilden, durch den man den Generator begrifflich äquivalent ebenfalls darstellen kann.

Redundanz ist Reglement. Reglement als Gesetzlichkeit, als ein typisches Wiederholungsmuster, kann selbst neuartig, originell sein. Zwei Beispiele aus der Geschichte der Malerei sind das berühmte schwarze Quadrat von Malewitsch und ein Bild *Zwei Akzente* von Max Bill, das zwei gleichartige Zeichen, farblich und lagemäßig voneinander unterschieden, übereinander anordnet. Bei Malewitsch ist die Herausstellung origineller Redundanz durch die Symmetrie des schlichten Quadrats auf die Spitze getrieben. Kehren wir nun zu unseren Bildgeneratoren zurück, so sehen wir, daß auch hier die auf ästhetischen Vorentscheidungen des Programmierers beruhende Bildredundanz innovativen Charakter hat. Der Innovationstyp, auf den wir hier stoßen, repräsentiert in einem nicht scharf definierbaren Sinn die Thematik des Bildes, wie sie der Intention des Programmierers entspricht.

Die ästhetische Redundanz im Produkt eines Bildgene-

rators birgt also eine bestimmte Art von Innovation. Da diese Innovation nichts mit der von der Funktion der Zufallsgeneratoren abhängigen Mikroinnovation zu tun hat, müssen wir sie besonders kennzeichnen. Zunächst können wir sie unter den allgemeinen Begriff der Makroinnovation subsummieren, den wir hiermit einführen. Makroinnovation bezeichne alle diejenigen innovativen Bildanteile, die zur ästhetischen Information des Bildes beitragen, jedoch ohne Rückgang auf die mikroästhetischen, mikroinnovativen Bilddeterminanten schlicht apperzipiert werden können. Dieser Definition genügt die in der ästhetischen Redundanz erscheinende Innovation ohne Zweifel. Da Innovation dieser Art von außen stammt, durch den Schöpfer und Betreiber des Bildgenerators induziert wird, heiße sie induzierte Makroinnovation. Im nächsten Abschnitt werden wir sehen, daß die Gesamtinnovation einer ästhetischen Information mehr enthalten kann, als nur Mikroinnovation und induzierte Makroinnovation.

3.1.6. Induzierte und inhärente Innovation

Treffen wir Innovation im Produkt eines Bildgenerators an, die wir nicht als vom Programmierer induziert erkennen können, so bezeichnen wir sie künftig als inhärent. Inhärent ist sicher alle Mikroinnovation. Als inhärent innovativ müssen jedoch auch diejenigen Bildeigentümlichkeiten angesprochen werden, die makroästhetische Auswirkungen mikroästhetischer Ursachen sind. Innovation mit makroästhetischen Effekten auf mikroästhetischer Basis heiße inhärente Makroinnovation. Ein typisches Beispiel für das Auftreten von inhärenter Makroinnovation bietet das untere Teilbild von Bild 17. Schon in Abschnitt 3.1.3 stellten wir fest, daß diese ästhetische Information durch endogene Beziehungen ange-

reichert ist, die durch die Überschiebung von Bildelementen zustande kommen. Die durch den Überschiebungs- oder Überlagerungseffekt zustandegebrachten neuen endogenen Bildanteile, die meistens den Charakter von Beziehungen und Verknüpfungen zwischen sonst eigenständigen Bildkomponenten haben, sind wegen der Autarkie der Funktion der Zufallsgeneratoren nur ungenau oder aber überhaupt nicht vom Programmierer vorhersehbar und planbar.

Einem späteren Abschnitt vorausgreifend ist zu sagen, daß inhärente Makroinnovation sowohl durch großräumige Bildeffekte als auch durch stabilisierende Komponentenverknüpfung gestaltschaffend wirkt. Im Sinne eines ganz naiven Gestaltbegriffes sind z. B. die Inhalte der Bilder 19 und 20 Gestalten. In ihrer eigenständigen Individualität makroästhetisch vor uns stehend, sind diese Gestalten jedoch konstituiert von den Zufallsgeneratoren und sind deshalb als Individuen absolut unplanbar. Induzierte Makroinnovation ist hier lediglich das Wissen, daß diese Gestalten durch die geradlinige Verbindung eines Systems von Stützpunkten zustande kommen, aber diese allgemeine Information begründet kein gestalthaftes Individuum.

3.1.7. Signal, Rauschen und Mikroinnovation

Signale, als die Träger von Nachrichten, d. h. als Substrat von Information, sind zwar technisch gemacht, folgen aber physikalischen Gesetzen. Die Nachrichtentechnik hat Interesse daran, Signale möglichst unverfälscht von ihrem Ursprungs- zu ihrem Bestimmungsort zu bringen. Das Medium, das die Signale und die von ihnen getragenen Nachrichten durchlaufen, bezeichnet die Nachrich-

tentechnik und die Informationstheorie als Nachrichtenkanal oder Informationskanal oder einfach als Kanal. Nun ist es eine innere Eigenschaft aller technisch realisierbaren Kanäle, die Information, die sie befördern, zu stören. Information und ihre Trägersignale gelangen also gewöhnlich nicht unverfälscht beim Informationsperzipienten an. Kanäle stören Information dadurch, daß sie von sich aus Information zur beförderten Information addieren, die beförderte Information also durch eine Störinformation überlagern. Diese Störinformation nennt man das Rauschen des Kanals.

Vom Standpunkt der Nachrichtentechnik aus betrachtet, ist also das Rauschen eine zwar unvermeidliche, jedoch negativ bewertete Eigenschaft von Kanälen. Nun ist es zunächst interessant, daß die Ausdrücke *Kanal* und *Transduktor* synonym gebraucht werden (Ashby, a. a. O., Seite 154). Auch jeder Transduktor befördert ja Information; er nimmt Eingangsinformation auf und gibt Ausgangsinformation ab. Die Ausgangsinformation ist zwar eine Transformierte, ein Abbild der Eingangsinformation, eine Informationstransformation irgendwelcher Art findet aber in jedem Nachrichtenkanal statt (er wird beispielsweise Betätigungen von Tasten in Leuchtsignale verwandeln). Es liegt deshalb nahe, die mit Zufallsgeneratoren versehenen Transduktoren des vorhergehenden Abschnitts als verrauschte Kanäle zu betrachten. Verrauschend wirkt die Tätigkeit der Zufallsgeneratoren, weil sich die jeweilige Größe der von ihnen im Einzelakt hervorgebrachten Zufallszahl der Kontrolle des Programmierers entzieht. Denn wir haben nie die Möglichkeit methodologisch eingeschlossen, der Benutzer des Kanals könnte die Wirkung der Zufallsgeneratoren absichtlich berechnen, um die Ausgangsinformation des Transduktors restlos zu kennen und steuerbar zu ma-

chen. Eine derartige Motivation — und das ist der Kern der Sache — findet sich im Rahmen der generativen Graphik nicht.

Unsere Transduktoren sind also verrauschte Kanäle. Jedoch bewerten wir im Gegensatz zu den Nachrichtentechnikern die Verrauschung nicht negativ, sondern positiv. Wir sind gar nicht daran interessiert, den Störeffekt des Kanalrauschens systematisch zu erfassen, um ihn zu eliminieren, im Gegenteil haben wir die Zufallsgeneratoren programmatisch in die Sprache G für die Beschreibung von Geneseprozessen eingebaut. Man tritt also in eine relativ verkehrte Welt ein, wenn man den Bereich der generativen Ästhetik betritt. Fassen wir es zusammen: Ein Kanal der Nachrichtentechnik nimmt eine Serie von Nachrichten als Eingangsinformation auf und liefert diese Serie harmlos transformiert, bedauerlicherweise aber auch durch Rauschen gestört, am Kanalende als Ausgangsinformation ab. Ein Transduktor der generativen Graphik nimmt einen Satz von Parametern als im Verlauf des nachfolgenden Prozesses konstant gehaltene Eingangsinformation auf und liefert eine Serie von Transformierten der Eingangsinformation am Ausgang des Transduktors ab, wobei jede der Transformierten erfreulicherweise durch die Einwirkung der Zufallsgeneratoren um eine additive Information bereichert wird. Bei den Nachrichtenkanälen ist die additive Information Störinformation, bei unseren Transduktoren ist sie erwünschte Mikroinnovation.

3.2. Gradationen

Eine der erregendsten Tatsachen in der Welt der Zeichenkonstellationen ist die Existenz vollkommen ver-

schiedener Modi ästhetischer Information. Wenn man beispielsweise die Gestalt eines Baumes aus angemessener Entfernung betrachtet, so kann man durchaus so motiviert sein, daß man der Personalisierung des gesehenen Objekts nicht entgeht. Der Baum wird dann zu diesem starken, ehrwürdigen, trotzigen, verträumten, in sich ruhenden usw. Nähert man sich dem Baum so weit, daß man nur noch die Rinde im Blickfeld hat und betrachtet man sie aus rein ästhetischem Interesse, so schaltet die Wahrnehmung um und man nimmt jetzt in einer Weise wahr, die auch beim Umsichblicken in einer gleichförmig gewellten Wüstenfläche herrscht, in der man allein ist. Im vorliegenden Hauptabschnitt widmen wir uns der modellmäßigen Untersuchung und Unterscheidung solcher verschiedenen Informationsqualitäten; zur Vorbereitung betrachte man etwa nacheinander die Bilder 52 und 30, oder auch 51 und 49.

Ein zweites gilt: Wüstenlandschaften z. B. können ganz verschieden strukturiert sein. Es gibt völlig glatte, gerippelte, gewellte, solche mit eingestreuten Steinblöcken, mit Hügelzügen, den Hügelzügen können Felsformationen aufgesetzt sein, schließlich kann die Wüste gebirgig und dadurch stark gegliedert sein. Was wir eben aneinandersetzten, war eine Skala verschiedener Formen, die doch alle zu einer, nämlich einer nicht-gestaltorientierten Informationsklasse zählen. Derartige Skalen nennen wir Gradationen. Eine Gradation bilden z. B. die Bilder 49 bis 51.

Das Anliegen dieses Hauptabschnittes ist der Versuch einer Zerfällung der Mannigfaltigkeit ästhetischer Informationen in qualitativ verschiedene Gradationen. Auf diese Weise werden wir das Klassifikationsproblem der generativen Graphik angehen.

3.2.1. Ein Rahmenprogramm für eine Experimentalserie

SERIE(QUER,HOCH,XMAL,YMAL,FIGUR) ist eine Prozedur, die ein achsenparalleles Rechteck in XMAL mal YMAL kongruente Elementarrechtecke der Abmessungen QUER Längeneinheiten entlang der X-Achse und HOCH Längeneinheiten entlang der Y-Achse einteilt. SERIE zeichnet also im allgemeinen nicht die Abgrenzungen der Elementarrechtecke, vielmehr zeichnet SERIE in den Rahmen, der jeweils durch ein Elementarrechteck virtuell abgegrenzt wird, eine Elementarfigur ein, deren Aufbau durch eine Prozedur FIGUR durchgeführt wird. Der Formalparameter FIGUR in SERIE muß bei jedem Aufruf dieser Prozedur durch eine aktuale Prozedur ersetzt werden. Ist etwa S eine solche Aktualprozedur, so bestimmt S bei einem Aufruf von SERIE in jedem Elementarrechteck eine Elementarfigur. Zur Positionierung der Elementarfigur dienen dabei zwei globale Größen P und Q, die Translationen entlang der X- bzw. Y-Achse veranlassen.

Die Vereinbarungsfolge (1) enthält auch die Vereinbarung der Prozedur SERIE. Vorauslaufend werden in den Programmzeilen 1 bis 6 nützliche globale Größen vereinbart, darunter die schon erwähnten Positioniergrößen P und Q. SERIE selbst ist außerordentlich einfach aufgebaut. Diese Prozedur gruppiert Elementarrechtecke um den Koordinatenursprung herum und bestimmt deshalb zunächst in den Zeilen 15 und 16 aus den Aktualwerten von QUER, XMAL, HOCH und YMAL die Anfangswerte von P und Q, d. h. die Lage des allerersten Elementarrechtecks der Serie, das grundsätzlich in den dritten Quadranten des Koordinatensystems zu liegen kommt.

```
(1)   1  'REAL' P, Q, P1, Q1
      2  ,XM, YM, HOR, VER
```

```
 3  ,JLI, JRE, JUN, JOB
 4  .,
 5  'INTEGER' I, M, N, T
 6  .,
 7  'PROCEDURE' SERIE
    (QUER,HOCH,XMAL,YMAL,FIGUR).,
 8  'VALUE' QUER, HOCH, XMAL, YMAL.,
 9  'REAL' QUER, HOCH.,
10  'INTEGER' XMAL, YMAL.,
11  'PROCEDURE' FIGUR.,
12  'BEGIN'
13  'REAL' YANF.,
14  'INTEGER' COUNTX, COUNTY.,
15   P.=−QUER*XMAL*.5.,
16   Q.=YANF.=−HOCH*YMAL*.5.,
17  'FOR' COUNTX.=1 'STEP' 1 'UNTIL' XMAL
    'DO'
18  'BEGIN' Q.=YANF.,
19  'FOR' COUNTY.=1 'STEP' 1 'UNTIL' YMAL 'DO'
20  'BEGIN' FIGUR., Q.=Q+HOCH
21  'END'., P.=P+QUER
22  'END'.,
23   LEER(−148.0,−105.0)., CLOSE.,
24   SONK(11).,
25   OPEN(X,Y)
26  'END' SERIE.,
```

Den wesentlichen Abschnitt von SERIE bilden die Zeilen 17 bis 22, die zwei ineinandergeschachtelte Laufanweisungen von genau der gleichen Bauart darstellen, wie wir sie schon zur Genese der Faltermatrix in Bild 1 benutzt haben (siehe Abschnitt 2.1.3).

Zeile 20 enthält die zentrale Stelle der ganzen Prozedur. Hier tritt der in Zeile 11 als Prozedur spezifizierte For-

malparameter FIGUR wieder auf. An dieser Stelle, im Laufkern der Doppelschleife, wird im Feld des jeweils von SERIE angesteuerten Elementarrechtecks diejenige Prozedur zur Ausführung gebracht, die beim Aufruf von SERIE den formalen Parameter FIGUR ersetzt; an die Stelle von FIGUR könnte also z. B. die weiter oben erwähnte Aktualprozedur S treten. Wir haben hier den Fall vor uns, dessen Möglichkeit wir schon in Abschnitt 2.1.6 erwähnten, daß nämlich Prozedurrümpfe Aufrufe weiterer Prozeduren enthalten dürfen. Im Fall des Auftretens von FIGUR in SERIE ist die aufgerufene Prozedur prinzipiell parameterlos.

Eine geringfügige Erweiterung der Sprache G trifft man in Programmzeile 24 an. SERIE ist nämlich auf die automatische Hintereinanderschaltung von Reihen von Serienbildern insofern eingerichtet, als der Prozeduraufruf von LEER in Zeile 23 den Zeichenautomaten selbsttätig zur linken unteren Ecke des DIN-A 4-Zeichenblatts zurückbeordert. Zeile 24 ist „Sonderkommando Nr. 11" zu lesen und stellt eine Anweisung in der Form des Aufrufs einer horizontglobalen Prozedur SONK(N) dar. Im Fall SONK(11) löst SONK einen Zwischenstop zwischen dem Zeichnen von zwei Serien aus, der dazu dienen kann, die Zeichenblätter und Zeichenstifte auszuwechseln. In Zeile 25 von (1) wird das Zeichnen zum Zweck der Bearbeitung des nächsten Bildes sofort wieder neu eröffnet.

Ein Programmierbeispiel wird die Funktionsweise der Prozedur SERIE am besten veranschaulichen. Wir wählen ein Elementarrechteck der horizontalen Ausdehnung HOR = 65 Längeneinheiten und der vertikalen Ausdehnung VER = 20 Längeneinheiten. HOR und VER sind also Aktualparameter, die später beim Aufruf von SE-

RIE für die Formalparameter QUER und HOCH eingesetzt werden können. Den in (1), Zeile 5, vereinbarten Zähler M benutzen wir dazu, die Elementarrechtecke in der Reihenfolge abzuzählen, in der sie nacheinander von der Prozedur SERIE angesteuert werden. Nun schreiben wir eine Prozedur S, die später an die Stelle von FIGUR tritt. S zeichnet zunächst den Umriß eines Rechtecks, dessen Abmessungen um eine Längeneinheit geringer sind, als die des Elementarrechtecks. Danach zeichnet S in das schon gezeichnete Rechteck so viele vertikale Striche ein, als durch den momentanen Stand des Zählers M angegeben wird. Folgendermaßen ist der Aufbau von S:

(2)

```
 1 'PROCEDURE' S., 'BEGIN'
 2 'INTEGER' D.,
 3  LEER(P+1,Q+1)., LINE(P+HOR-1,Q+1).,
 4  LINE(P+HOR-1,Q+VER-1).,
 5  LINE(P+1,Q+VER-1)., LINE(P+1,Q+1).,
 6 'FOR' D.=1 'STEP' 1 'UNTIL' M 'DO'
 7 'BEGIN'  LEER(P+1+D,Q+1).,
 8            LINE(P+1+D,Q+VER-1)
 9 'END'.,
10  M.=M+1
11 'END' S.,
```

in den Programmzeilen 3, 4, 5, 7 und 8 sieht man deutlich, wie die Variablen P und Q zur örtlichen Positionierung der Figur S benutzt werden. Das korrekte Weiterschalten der Werte von P und Q von Elementarrechteck zu Elementarrechteck besorgt die Prozedur SERIE mit Hilfe ihrer zwei ineinandergeschachtelten Laufanweisungen. Da SERIE weiter nichts zu tun hat, ist der Aufbau dieser Prozedur so einfach.

Zwei Programmzeilen genügen zur Herstellung von Bild 21:

```
(3)  1  M.=1., HOR.=65., VER.=20.,
     2  SERIE(HOR,VER,4,7,S).,
```

Das Bild zeigt, wie SERIE, aktualisiert durch den Prozeduraufruf in Zeile 2 von (3), die Figur S in die Elementarrechtecke des Flächenmaßes HOR mal VER verteilt. Das Anwachsen der Anzahl der vertikalen Striche in der Einzelfigur S zeigt die Reihenfolge an, in der SERIE die 28 Elementarrechtecke ansteuert: Es geschieht spaltenweise, die Spalten entlang der X-Achse reihend. Bild 21 ist rein deterministisch, Zufallsprozesse fehlen.

3.2.2. Der elementare Irrweg

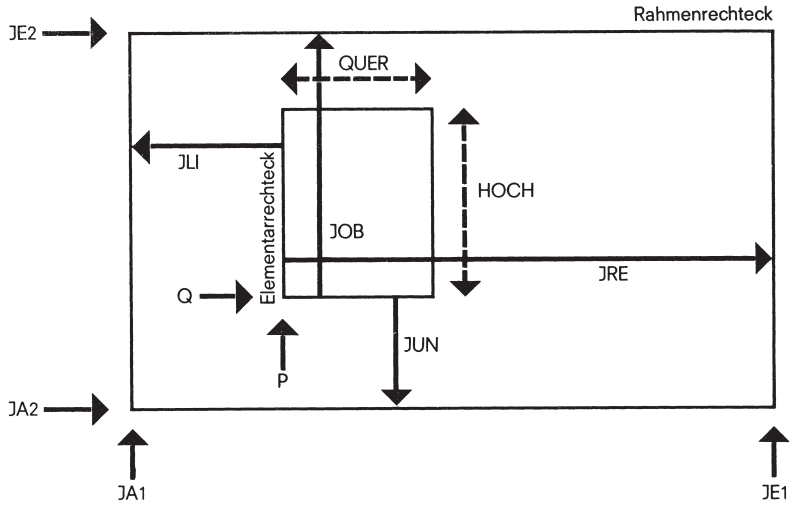
Nachdem wir jetzt im Besitz der Mehrzweckprozedur SERIE sind, standardisieren wir die figurenzeichnende Prozedur, die den Formalparameter FIGUR in SERIE zu ersetzen hat. Die parameterlose Prozedur ELIRR zeichnet einen zufälligen Irrweg in ein Rahmenrechteck ein, dessen Eckpunktkoordinaten mit den Intervallgrenzen der Zufallsgeneratoren J1 und J2 übereinstimmen. Der durch J1 und J2 gesteuerte Irrweg besteht aus schrägen oder achsenparallelen Teilstrecken, je nachdem, ob die in Zeile 5 der Vereinbarungsfolge (1/3.2.1) vereinbarte ganzzahlige und in ELIRR globale Größe T den Wert 1 oder 0 hat.

```
(1)  1  'PROCEDURE' ELIRR.,
     2  'BEGIN'
     3  JA1.=P+JLI., JE1.=P+JRE.,
     4  JA2.=Q+JUN., JE2.=Q+JOB.,
     5  P1.=J1., Q1.=J2.,
```

Bild 21

```
 6  LEER(P1,Q1).,
 7  'FOR' M.=1 'STEP' 1 'UNTIL' N 'DO'
 8  'IF' T 'EQUAL' 0 'THEN'
 9  'BEGIN'
10   LINE(J1,Y)., LINE(X,J2)
11  'END'
12  'ELSE' LINE(J1,J2).,
13  'IF' T 'EQUAL' 0 'THEN'
14  'BEGIN'
15   LINE(P1,Y)., LINE(X,Q1)
16  'END'
17  'ELSE' LINE(P1,Q1)
18  'END' ELIRR.,
```

In (1) dienen die Zeilen 3 und 4 zur Koordinierung des jeweiligen Elementarrechtecks, das von SERIE angesteuert wird, mit dem Rahmenrechteck, in das ELIRR seinen Zufallsirrweg einbeschreibt. Die beiden Rechtecke fallen genau dann zusammen, wenn die Aktualwerte der Parameter QUER und HOCH von SERIE mit den Werten der globalen Größen JRE und JOB übereinstimmen und wenn die ebenfalls globalen Größen JLI und JUN den Wert null haben. In allen anderen Fällen ergibt sich eine gewisse Verschiebung in der Lage der im übrigen achsenparallelen Rechtecke, die durch die Zeilen 3 und 4 von (1) bewirkt wird. Figur 5 zeigt die gegenseitige Abhängigkeit der Größen JA1, JE1, JA2, JE2, QUER, HOCH, JLI, JRE, JUN, JOB, P und Q. Die beiden zuletzt genannten Größen fixieren die Momentanlagen der durch SERIE und ELIRR seriell determinierten Rechteckpaare.



Figur 5

Wir fahren mit der Analyse der Prozedur ELIRR fort. Die Zeilen 5 und 10 zeigen den Aufruf der zwei Zufallsgeneratoren J1 und J2, denen vor dem ersten Aufruf von ELIRR Anfangswerte zugewiesen werden müssen. Eine gewisse Schwierigkeit taucht in den Zeilen 12 und 17 auf, nämlich das Wortsymbol 'ELSE'. Dieses Wortsymbol ist Teil einer Weichenkonstruktion, die wir bis jetzt nicht benutzt haben und die wir daher besprechen müssen. Wir erklären die 'IF'—'THEN'—'ELSE'-Weiche in der gleichen Weise, wie wir in Abschnitt 2.1.4 die einfache 'IF'—'THEN'-Konstruktion einführten. In

(2) 'IF' aussage 'THEN'
anweisung1 'ELSE'
anweisung2., anw.3.,

vertreten anweisung1, anweisung2 und anw3 beliebige Anweisungen. Der syntaktische Typ aussage repräsentiert wie in (1/2.1.4) eine Aussage. Umgangssprachlich kann (2) folgendermaßen ausgedrückt werden: *Ist die Aussage aussage wahr, dann führe die Anweisung anweisung1 aus. Ist dagegen die Aussage aussage falsch, dann führe die Anweisung anweisung2 aus. Führen nicht Sprungbefehle innerhalb von anweisung1 oder anweisung2 an andere Stellen des Programms, so führe nach anweisung1 oder anweisung2 die Anweisung anw3 aus.* Als vollständige Dichotomie ist die Konstruktion (2) bequem zu handhaben, sie ist jedoch redundant, da sie auf die primitivere Anweisungsfolge

(3) 'IF' aussage 'THEN'
'BEGIN' anweisung1., 'GOTO' marke
'END'.,
anweisung2., marke.. anw3.,

reduziert werden kann.

Im Rumpf der Prozedur ELIRR veranlassen Weichen der Art (2) die Auswahl der durch den Wert von T gesteuerten Möglichkeiten, einen stückweise schräg oder achsenparallel verlaufenden Irrweg zu zeichnen. Übrigens ist der von ELIRR aufgebaute Irrweg ein geschlossenes Polygon, wofür die Zeilen 13 bis 17 der Prozedur (1) sorgen. Die Anzahl der Ecken des geschlossenen Polygons wird durch den Wert der globalen Variablen N in Zeile 7 der Prozedur ELIRR bestimmt. Ist K die Anzahl der Ecken, so ist K=N+1 im Fall T=1 und K=2*(N+1) im Fall T=0. Die Eckenanzahl K vermindert sich in der Bildinformation, wenn die endliche Dicke des Zeichenstiftes eng benachbarte Ecken zusammenfallen läßt.

3.2.3. Variationen

Nach langwierigen Vorbereitungen treten wir wieder in den eigentlichen Bereich der Ästhetik ein. Wir sind nun im Besitz der Prozedur SERIE, die Figuren auf einem Schachbrettuntergrund anordnet. Es ruft SERIE die Prozedur ELIRR auf, wobei ELIRR auf jedem (quadratischen oder rechteckigen) Feld des Schachbretts eine Figur anbringt, die ihrer Struktur nach ein Irrwegpolygon ist. Was läßt sich mit diesem Mechanismus anfangen?

Bei geeigneter Wahl der in Figur 5 auftretenden Größen liefert uns die Kombination von SERIE mit ELIRR zweifellos eine Figurenserie, d. h. eine Figurenanordnung mit gegenseitig voneinander abgesetzten Einzelelementen. Die Bilder 22 und 23 zeigen zwei Versuchsergebnisse, die durch den Ablauf eines Programms folgenden Aufbaus zustandekommen:

(1)

```
 1 'BEGIN'
 2 'REAL' P, Q, P1, Q1 . . .
 3 'PROCEDURE' SERIE . . .
 4 'PROCEDURE' ELIRR . . .
 5 'END' ELIRR.,
 6 'BEGIN' 'COMMENT' VARIATIONEN AUS
 7                   IRRWEGRECHTECKEN
 8                   ACHSENPARALLEL
                     UND SCHRAEG.,
 9 JI1.=JS1., JI2.=JS2.,
10 N.=20.,
11 JLI.=0., JRE.=20.,
12 JUN.=0., JOB.=20.,
13 T.=0.,
14 SERIE(20.0,20.0,8,6,ELIRR).,
15 T.=1.,
16 SERIE(20.0,20.0,8,6,ELIRR).,
17 'END' VARIATIONEN.,
```

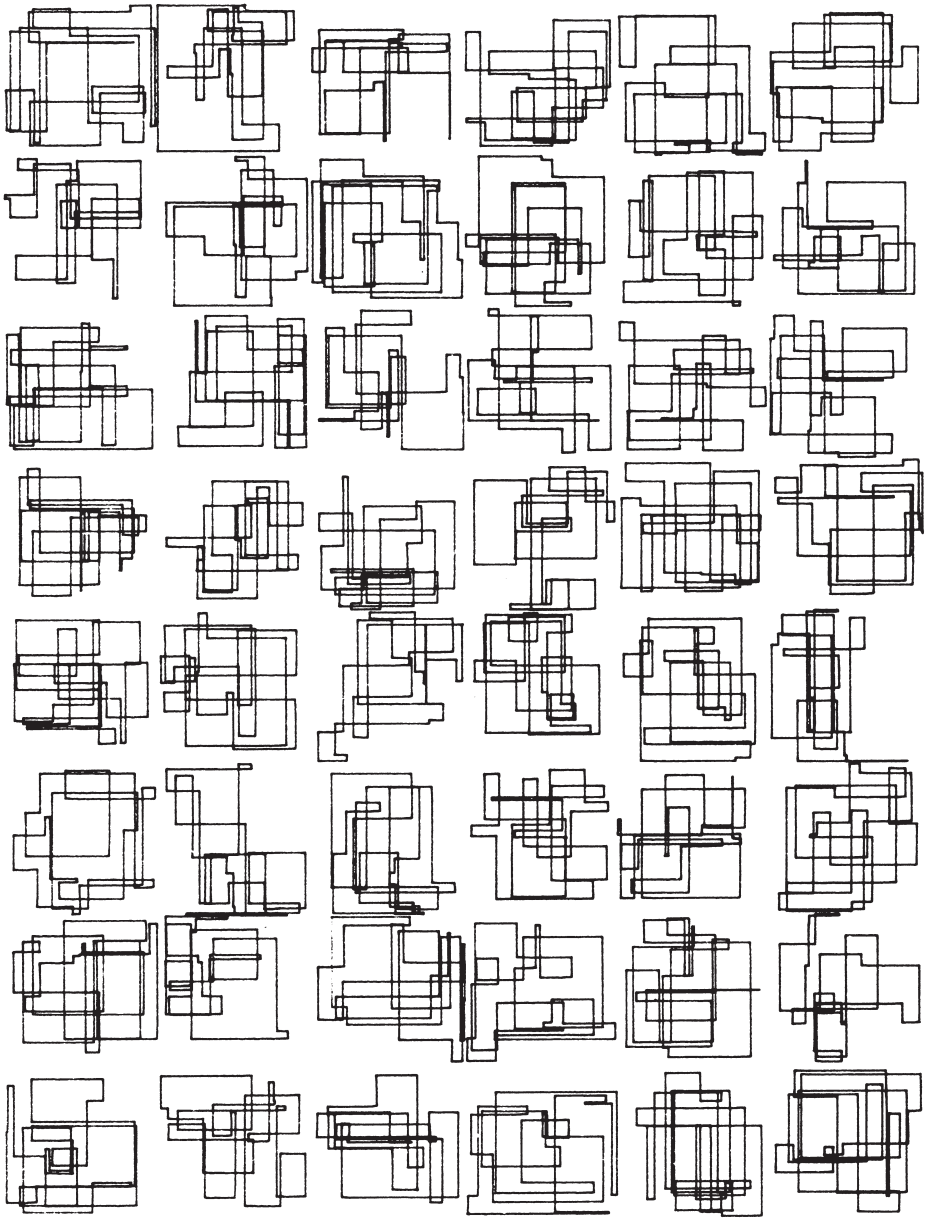
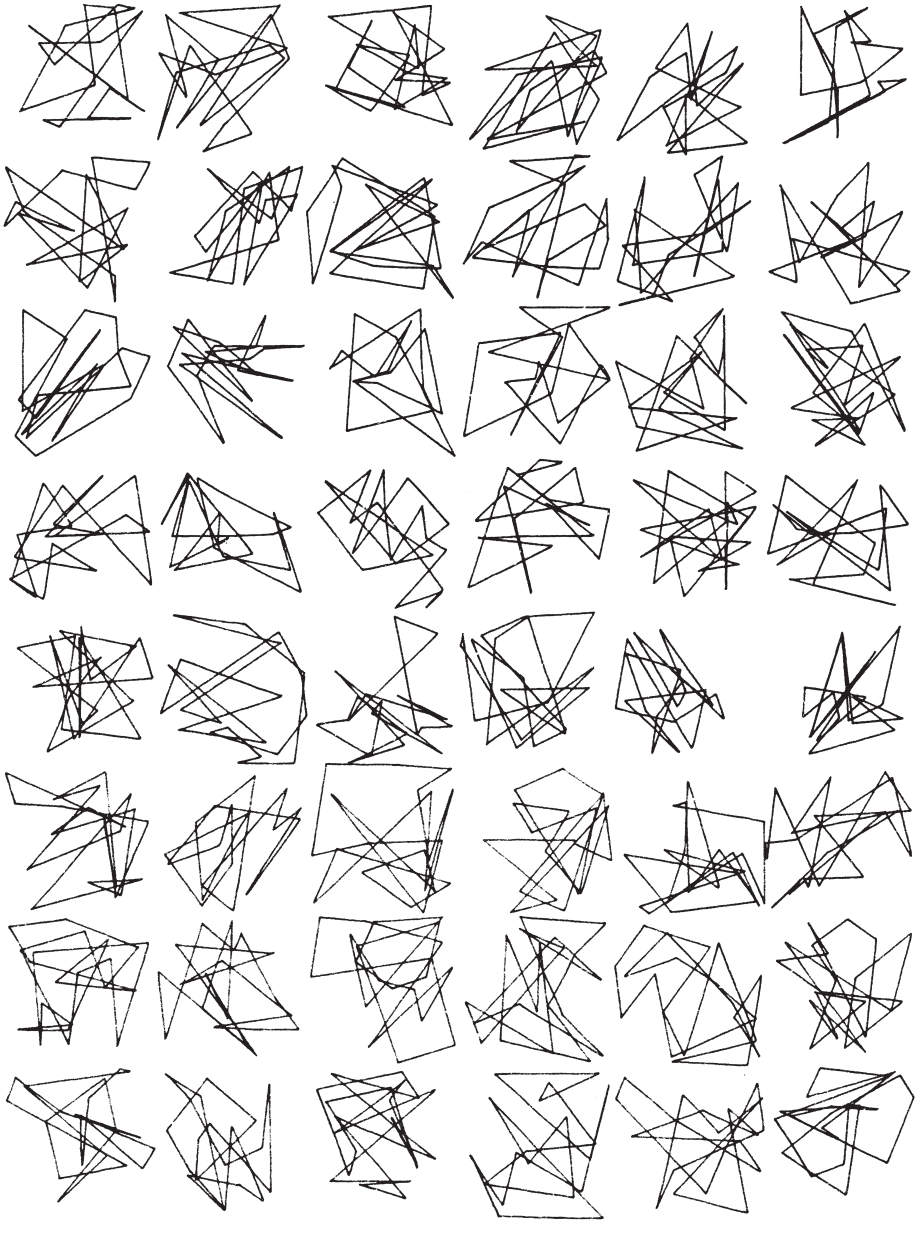


Bild 22

Bild 23



Da der Aufbau der Prozeduren SERIE und ELIRR schon erläutert wurde, genügt in den Zeilen 3 und 4 die Andeutung bekannter Programmteile durch „. . .". Von Zeile 6 bis 17 reicht der Generator für die Bilder 22 und 23. Nach den vorbereitenden Zeilen 9 bis 12 determiniert Zeile 13 Bild 22 als aus achsenparallelen Teilstrichen bestehend. Analog bewirkt Zeile 15 schräge Teilstriche. Die eigentlichen Aufrufe von SERIE in den Zeilen 14 und 16 sind völlig gleich gebaut. Der erste Aufruf löst die Genese von Bild 22, der zweite die von Bild 23 aus.

Man beobachtet, daß die von ELIRR generierten Einzelfiguren sich in Bild 23 besser voneinander abheben, als in Bild 22, obgleich die Bereichsgrenzen für den Irrweg in beiden Bildern die gleichen sind. Der Grund liegt in der größeren Ausladung oder Konvexität der achsenparallel aufgebauten Einzelfiguren von Bild 22. Jedem Schrägstrich im Fall T=1 entspricht ja ein Winkelstück mit achsenparallelen Schenkeln im Fall T=0, was sofort aus den Zeilen 12 bzw. 10 der Prozedur ELIRR hervorgeht (s. Vereinbarung 1/3.2.2). Übrigens ist es unmöglich, Eckpunktlagen in den Einzelfiguren der beiden Bilder 22 und 23 rezeptorisch miteinander korrespondieren zu lassen, da die Zufallsgeneratoren, bevor sie Bild 23 aufbauen, nicht neu initiiert werden. Die Zufallsgeneratoren fahren also nach der Genese von Bild 22 mit den Zufallszahlen fort, die in der Gesamtfolge von Zufallszahlen gerade in diesem Augenblick erscheinen; diese Zufallszahlen werden sofort beim Aufbau von Bild 23 verwendet. Man erhielte aus dem gleichen Grund eine Variante von Bild 23 mit anderen Einzelfiguren, führte man die Anweisung in Zeile 16 von (1) ein zweites Mal aus.

Solche Systeme von Einzelfiguren, die voneinander abgesetzt frei dastehen, jedoch in matrizenartiger Anord-

nung manifest sind, bezeichnen wir als Variationen. Das Wort *Variation* meint sowohl den Prozeß des Durchvariierens, als auch dessen Ergebnis. Wählt man eine Ausdrucksweise aus dem Bereich der Musik und sagt man *Variation über ein Thema* anstatt *Variation,* so gewinnt man gleichzeitig einen Zugang zur Informationsästhetik insofern, als das Thema einer Variation die Redundanz in der Variation bestimmt. Da, wie wir in Abschnitt 3.1 gelernt haben, Dispersion eine informationszerstreuende Funktion ausübt, haben wir bei der Variation eine Zeichenverteilung vor uns, bei der sich die als Thema auffaßbare, induziert makroinnovative Redundanz in diskreten Einzelfiguren dispergiert wiederfindet. Von verschiedenen Prozeßmodi, wie sie in AE, Seite 141, aufgezählt werden, nämlich: Destruktion, Dekomposition, Modulation, Symbolisierung, ist besonders die Modulation am Zustandekommen der Variation beteiligt. Ästhetische Information erscheint moduliert als eine Serie von Moduln, wobei der Modulationsprozeß nichts anderes als eine Weise des generellen Dispersionsprozesses ist. Jedoch auch die anderen Modi wirken in einer Art, die den ästhetischen Typ der Variation kennzeichnet: Destruktiv ist der Variationsprozeß insofern, als er den kontinuierlichen Nexus innerhalb des Rahmens der Variation, als einer Zeichenverteilung, weitgehend zerstört: Von den je achtundvierzig Einzelfiguren in den Bildern 22 und 23 sind benachbarte sich so unähnlich oder ähnlich, wie weit voneinander entfernte. Die Diskretion der Einzelfiguren bedeutet eine Dekomposition, eine Abschwächung des kompositiven Funktionsmodus. Jede Einzelfigur ist ein sichtbares Symbol des thematischen Gehalts der Variation, jedoch die Matrix der Einzelfiguren erleichtert höchstens durch die Neutralität ihrer Struktur das Hervortreten von Innovation und Redundanz, ohne informativ an ihnen beteiligt zu sein.

Da wir generative Ästhetik treiben, taucht sofort die Frage auf, ob die Struktur des Bildgenerators einer Variation etwas mit den vorhin erwähnten Typeigenschaften der Variation zu tun hat. Das ist aber tatsächlich der Fall. Wir wissen ja, daß immer dann, wenn der Laufkern einer Schleife eines Zeichenprogramms an der Bildgenese wesentlich beteiligt ist, bei jedem Schleifendurchlauf ein gewisses Quant der Bildinformation generiert wird. Der Schleifenkern ist dabei der Ursprung von Redundanz, die durch den Schleifenmechanismus in den Bildbereich hineindispergiert wird. Dieser Dispersionsprozeß ist durch den übersichtlichen Aufbau des Variationsgenerators, den wir im Augenblick benutzen, exakt überschaubar: Schleifenkern ist die figurenerzeugende Prozedur ELIRR in der Doppelschleife von SERIE. Redundanz und gleichzeitig induzierte Makroinnovation bestimmend (siehe auch Abschnitt 3.1.6) sind die Abmessungen des Rahmenrechtecks, in das ELIRR den Irrweg einzeichnet, sowie die Schräge oder die Achsenparallelität der Irrwegkanten, schließlich die Anzahl der Irrwegkanten. Makroinnovation bestimmen also die Größen JLI, JRE, JUN, JOB, T und N. Die Mikroinnovation ist selbstverständlich Äußerung der Zufallsgeneratoren J1 und J2 in ELIRR.

An den Bildern 22 und 23 wird deutlich, mit welcher Vorsicht man vorgehen muß, wenn man gewisse Komponenten ästhetischer Information als makroästhetisch oder mikroästhetisch klassifizieren will. Die mikroinnovationsschaffende Funktionalität ist in beiden Bildern, ihrem verschiedenen Aussehen zum Trotz, völlig die gleiche. Man sieht das sofort, wenn man die Quellstellen der in den Bildern enthaltenen Mikroinnovation aufsucht, es sind dies die Zeilen 5, insbesondere jedoch 10 und 12 in der Prozedur ELIRR (siehe 1/3.2.2). In beiden Zeilen 10

und 12 werden die innovierenden Zufallsgeneratoren J1 und J2 aufgerufen, in beiden sogar in der gleichen Reihenfolge. Daß die erste Zeile Übereckverbindungen, die zweite Schrägverbindungen von Zufallspunkten erzeugt, ist eine Wirkung des globalen Baus des Bildgenerators und damit eine Induktion von Makroinnovation. Die mikroästhetische Verwandtschaft der Bilder 22 und 23 ist um so überraschender, als allein die Übereck- bzw. die Schrägverbindung den Feinbau eines komplizierten, mit Hilfe von ELIRR generierten Irrwegs zu bedingen scheint. Tatsächlich aber wird beispielsweise bei der Rezeption von Bild 22 die Feinstruktur der ästhetischen Information mit Hilfe einer Kombination verschiedener Innovationsmodi erfaßt: Mikroinnovation, inhärente Makroinnovation und induzierte Makroinnovation wirken zusammen. Besonders die inhärente Makroinnovation, die durch die Überschneidungen der Irrweglinien zustande kommt, trägt zur individuellen Information von Bild 22 bei. Entsprechendes gilt für Bild 23. Hier kommt eine Feststellung aus den AE (Seite 143) zur Geltung: „Selbstverständlich bleibt in dem ganzen Prozeß das hergestellte Kunstwerk als Ganzes die *ästhetische Observable,* von der auszugehen ist, und wir studieren die mikroästhetischen Zeichen und Vorgänge, die der unmittelbaren Wahrnehmung entzogen bleiben, an den makroästhetischen Effekten."

Variationenbilder machen den Prozeß der zyklisch wiederholten Eruption von Transduktoren besonders deutlich, den wir in Abschnitt 3.1.4 untersucht hatten. Wie wir uns erinnern, sind z. B. die Bilder 19 und 20 zwei sukzessive Eruptionen eines Transduktors. Man kann eine Variationenmatrix auffassen als Versammlung der aufeinanderfolgenden Eruptionen eines Transduktors auf der Zeichenfläche. Im Fall von SERIE ist ELIRR der

wiederholt aktive Transduktor. Alles, was in Abschnitt 3.1.4 über die aus einem gepulsten Transduktor hervorgehende Bildserie gesagt wurde, gilt auch hier: Die Varianten einer Variation ordnen sich mit einer gewissen Monotonie einem Typus unter und sind trotzdem — eine für die andere — wohlunterschiedene Individuen. Ihre Individualität verdanken sie dem Zufall.

3.2.4.
Die Variation zwischen Haufen und Gestalt

Schon im vorhergehenden Abschnitt schrieben wir den Variationenbildern eine destruktive Komponente insofern zu, als der Prozeß der Variation zusammen mit der Wirksamkeit der Zufallsgeneratoren den durchgehenden Nexus im Bild weitgehend zerstört. Dies gilt für Bild 23 stärker als für Bild 22, da sich in Bild 22 durch die starke räumliche Annäherung der Einzelfiguren ein durchgängiger Redundanzcharakter bemerkbar macht, der eine Zunahme der inhärenten Makroinnovation bedeutet. Die destruktive Komponente in Variationenbildern kann man dadurch abschwächen, daß man die Originalität der Einzelfiguren verstärkt, indem man übermäßige Redundanz aus ihnen entfernt. Beispielsweise läßt sich die Redundanz der vier gleichen Seiten des Rahmenquadrats für den Irrweg in beiden Bildern 22 und 23 vermindern, indem die eine Seite auffällig länger gemacht wird als die andere. Dies bedeutet einen Zuwachs an induzierter Makroinnovation. Das ganze Bild schließt sich zusammen und formiert eine dichte Reihung oder Gruppierung. Geht man von den Bildern 22 und 23 zu 24 und 25 über, so hat man ein Beispiel für diesen Vorgang. Die Bilder 24 und 25 werden durch den Generator

Bild 24

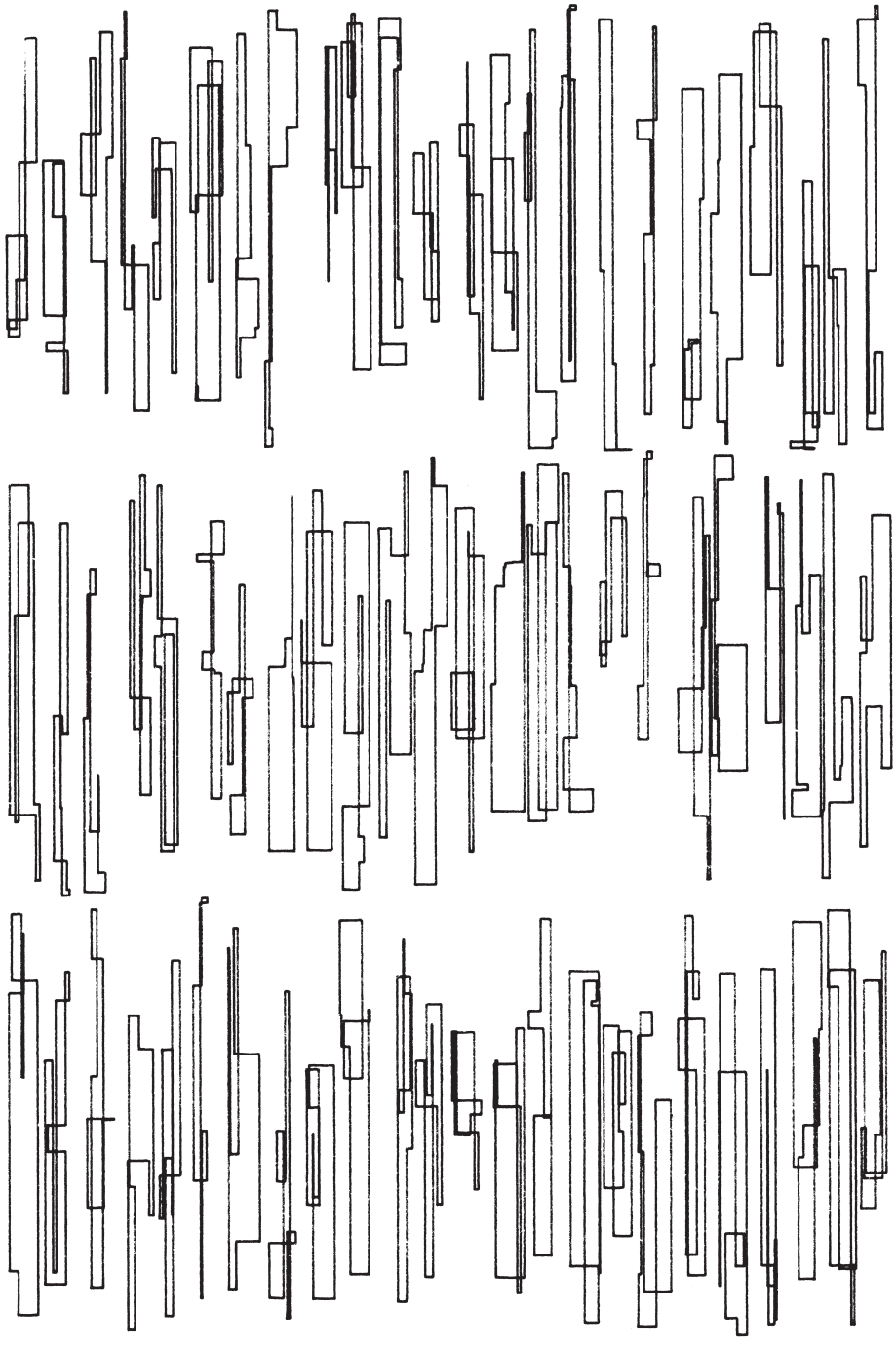
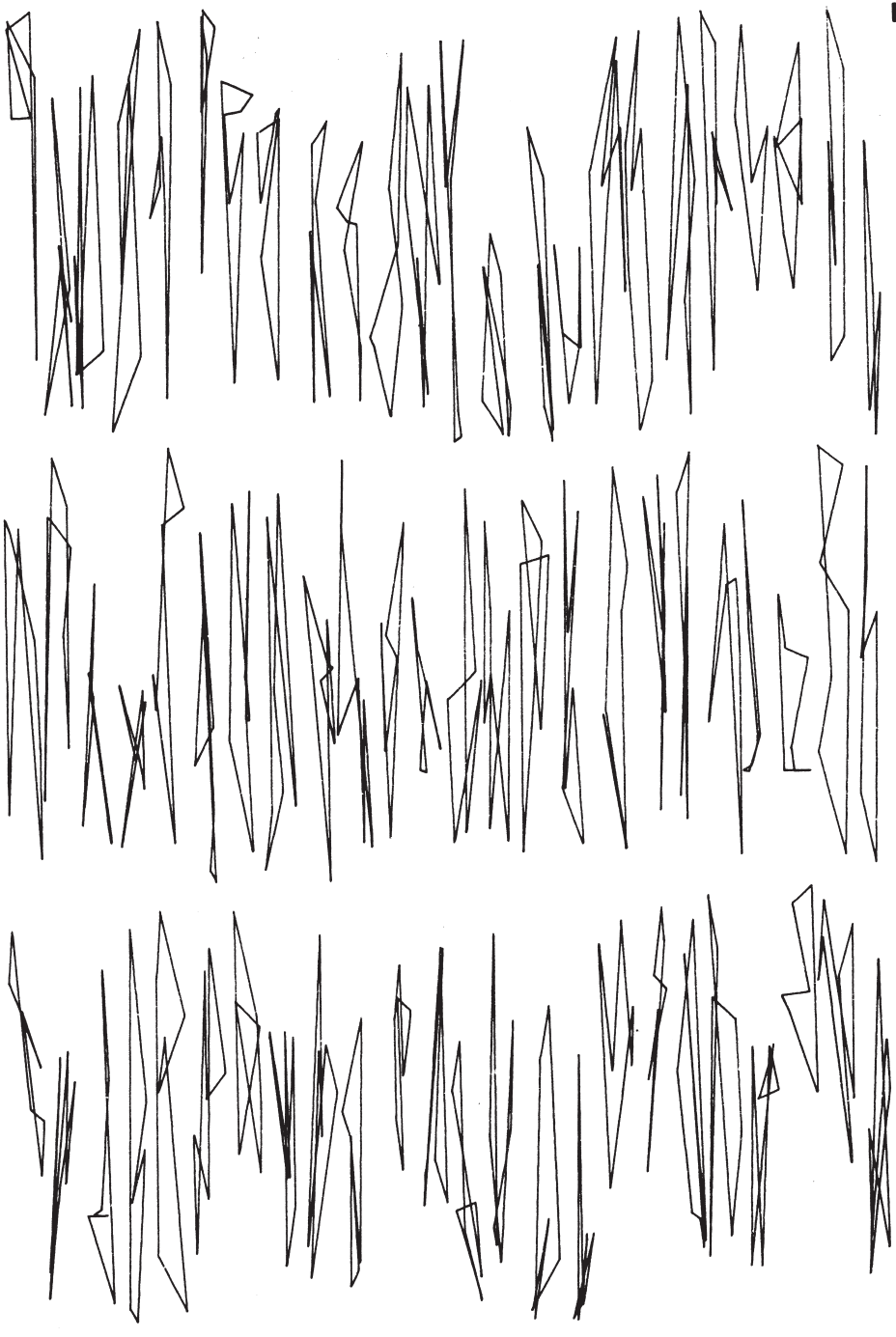


Bild 25



```
(1)     'BEGIN' 'COMMENT' ANISOTROPE SERIE.,
        JI1.=JS2., JI2.=JS5.,
        N.=6.,
        JLI.=0., JRE.=60.,
        JUN.=0., JOB.=5.,
        T.=0.,
        SERIE(60.0,5.0,3,24,ELIRR).,
        T.=1.,
        SERIE(60.0,5.0,3,24,ELIRR).,
       'END' ANISOTROPE SERIE.,
```

erzeugt, der die extreme Schlankheit der Einzelfiguren quantitativ belegt (die Aktualparameter 60 und 5 im Aufruf von SERIE stimmen mit den Kantenlängen des Rahmenrechtecks der Einzelfiguren überein). Wir bezeichnen die Übertreibung einer Ausdehnungsrichtung aller Einzelfiguren in Variationen als Anisotropie. Schrittweise Anisotropisierung eines Variationenbildes ist gleichbedeutend mit schrittweiser Steigerung der Makroinnovation des Bildes. Wir haben hier das erste Beispiel für eine Gradation vor uns. Wo endet diese Gradation? Die Frage ist natürlich nicht eindeutig zu beantworten. Bild 26 bietet ein Beispiel für einen möglichen Abschluß. Der zugehörige Generator ist

```
(2)     'BEGIN' 'COMMENT' EXTREMSERIE I.,
        JI1.=JS2., JI2.=JS3.,
        N.=40., T.=1.,
        JLI.=0., JRE.=240.,
        JUN.=10., JOB.=25.,
        SERIE(240.0,35.0,1,2,ELIRR)
       'END' EXTREMSERIE I.,
```

Man beachte nebenbei, daß wir gelegentlich die Initialwerte für die Zufallsgeneratoren wechseln. Dieser Wech-

Bild 26

sel ist dadurch begründbar, daß er hilft, das Einschleichen nichtzufälliger Parameter in die Tätigkeit der Zufallsgeneratoren zu vermeiden. In (2) werden von den sechs in (2/2.2.2) aufgeführten Initialwerten JS2 und JS3 verwendet.

Kehren wir nun zu der Frage nach der Art der Gradation zurück, die über die Bilder 23, 25 und 26 läuft! Im Sinn des naiven Gestaltbegriffs ist Bild 26 eine Gestalt. Das Bild kann als Doppel- oder Paargestalt bezeichnet werden, seine beiden Komponenten sind so eng aufeinander bezogen, daß eine voluntative Zerfällung durch den Perzipienten kaum möglich ist. Anisotropisierung, gleichbedeutend mit Zuwachs an Makroinnovation, führt also von Variationen zu Gestalten.

Zu ganz anderen Informationstypen gelangt man, wenn man wiederum von Variationen ausgeht, jedoch die Makroinnovation nicht verstärkt, sondern abschwächt. Eine Möglichkeit, die Makroinnovation zu vermindern, besteht darin, den strengen Matrizencharakter des Variationenbildes aufzuheben, ohne jedoch die Identität der Einzelfigur preiszugeben. Man bekommt auf diese Art und Weise Häufungen von Einzelfiguren: Haufenbilder oder Schwärme. Bild 27 ist ein Beispiel für einen Haufen, dessen Einzelfiguren Rechtecke sind. Das Bild ist wohlgemerkt nicht von dem uns wohlbekannten Transduktor SCHWARM (siehe (1/3.1.2)) erzeugt worden, sondern durch eine Anwendung von SERIE:

(3)

```
'BEGIN' 'COMMENT' HAUFEN
 AUS RECHTECKEN.,
'PROCEDURE' ELRECT.,
'BEGIN' 'REAL' D, E, F, G.,
JA1.=P+JLI., JE1.=P+JRE.,
```

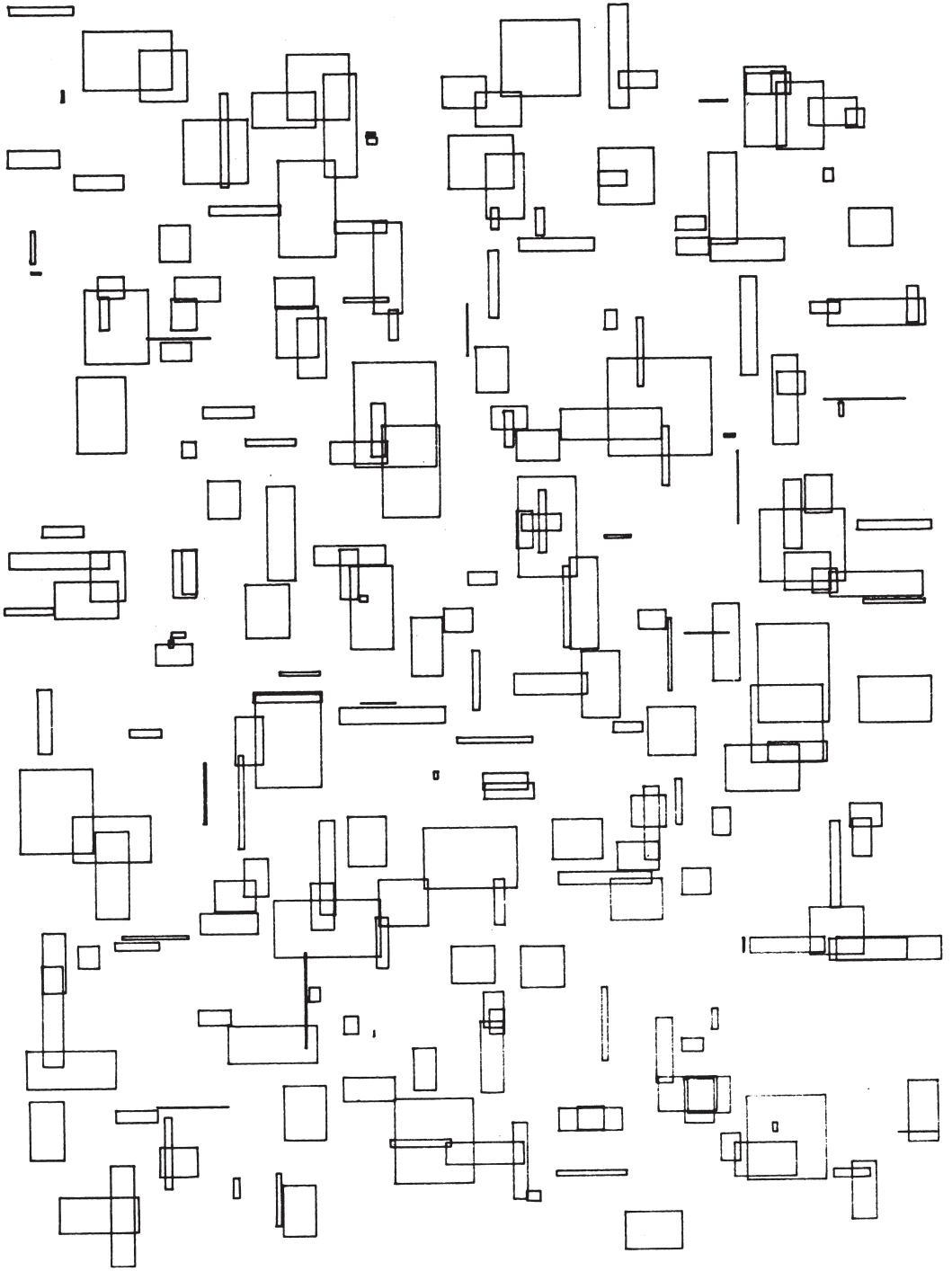
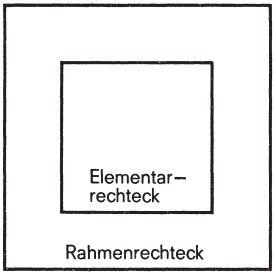


Bild 27

```
JA2.=Q+JUN.,JE2.=Q+JOB.,
D.=J1.,E.=J1.,
F.=J2.,G.=J2.,
LEER(D,F).,
LINE(E,F).,
LINE(E,G).,
LINE(D,G).,
LINE(D,F)
'END' ELRECT.,
JLI.=-5.,JRE.=15.,
JUN.=-5.,JOB.=15.,
SERIE(10.0,10.0,18,13,ELRECT)
'END' HAUFEN AUS RECHTECKEN.,
```

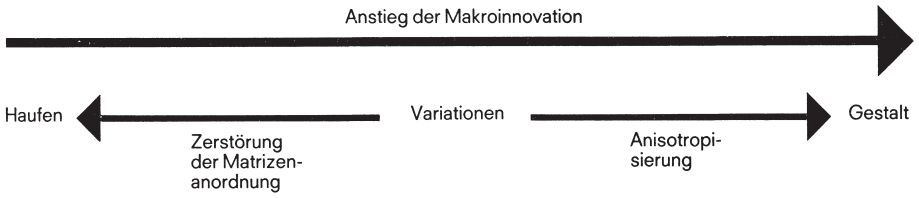
Für die Genese der einzelnen Rechtecke in Bild 27 wird ein besonderer, durch die Prozedur ELRECT repräsentierter Transduktor benutzt. Der Effekt der Haufenbildung kommt natürlich durch die gegenseitige Lage des Elementarrechtecks von SERIE und des durch die globalen Größen JLI, JRE, JUN, JOB bestimmten Rahmenrechtecks für die Einzelfiguren zustande. Figur 6 ist ein Spezialfall von Figur 5 und zeigt Lagerung und Proportionen von Elementarrechteck und Rahmenrechteck. Die Zerstörung der Matrizenanordnung ist dadurch bedingt,



Figur 6

daß die Rahmenrechtecke benachbarter Elementarrechtecke sich überlappen.

Wir haben gezeigt, daß eine unterbrechungslose Gradation von Haufen über Variationen zu Gestalten führt. Die Gradation ist mit ansteigender induzierter Makroinnovation verknüpft. Wie verhält sich die Mikroinnovation? Da sie in dem Beispiel einer Gradation vom Typus Haufen-Variation-Gestalt, den wir untersucht haben, unveränderlich durch die Zufallsgeneratoren J1 und J2 immer in der gleichen Weise bestimmt wird, könnte man sie als konstant bezeichnen. Dies wäre jedoch ein Trugschluß, da die Größe der Mikroinnovation — will man überhaupt ordnungstheoretische Aussagen machen — auch von Zeichenüberlagerungen und damit von der *lokalen Zeichendichte* im Bild bestimmt wird (siehe auch Abschnitt 3.2.6). Innerhalb der durch Figur 7 schematisierten Gradation kann also die Mikroinnovation mannigfachen Änderungen unterliegen.



Figur 7

Zum Schluß des vorhergehenden Abschnitts 3.2.3 erwähnten wir, daß sich das Variationenbild als die Versammlung der aufeinanderfolgenden Eruptionen eines Transduktors auf der Zeichenfläche auffassen läßt. In

Abschnitt 3.1.4 bezeichneten wir Eruptionenfolgen von Transduktoren als Superalphabete. Jedes Variationenbild stellt also gleichzeitig ein Zeichenalphabet dar (was zuerst Max Bense — mündlich dem Verf. gegenüber — bemerkte). Dieses Hervortreten eines Zeichenalphabets im Blickfeld des Perzipienten bedeutet jedoch, wie wir schon in Abschnitt 3.2.3 feststellten, eine Abschwächung des kompositiven Funktionsmodus in der dargebotenen ästhetischen Information. Komponierend wirkt nur noch der Matrizenrahmen als solcher, keine inhärenten Bildrelationen helfen mit. Geht man nun in der durch Figur 7 schematisierten Gradation nach links, so nimmt der kompositive Informationsanteil zweifellos noch weiter ab, und der Alphabetcharakter des Bildes verschwindet. Der Alphabetcharakter schwächt sich jedoch auch ab, wenn man die Gradation von der Mitte ausgehend nach rechts verfolgt, gleichzeitig wächst die kompositive Dichte des Bilds bis hin zur Gestalt. Auf jeden Fall setzt sich, wenn man von ganz links kommt, der kompositive Informationsanteil immer mehr durch, bis er in den Gestalten alle anderen Funktionsmodi überwältigt.

Innerhalb einer bestimmten Gradation des besprochenen Typs kann sich ein Anteil an induzierter Makroinnovation konstant durchhalten: Es ist dies ein makroinnovatives „Vor-Bild", das durch den innersten Schleifenkern im Generator einer Variation in die Variantenfolge dispergiert wird. Diese Innovation ist das Originelle, das alle Varianten gemeinsam haben. Als Beispiel diene eine Matrix von Rechtecken, die Bild 27 in der Gradation nach oben ergänzt. Enden möge die Gradation mit einem einzelnen Quadrat — womit sich das Vor-Bild herausschält. Nach einer physikalischen Analogie kann die Gradation von unten nach oben verfolgt als Kristallisation, umgekehrt gesehen als Verdampfung gedeutet werden.

3.2.5. Texturen

Die Variationen, die wir im vorangehenden Abschnitt besprochen haben, zerfallen in die voneinander abgesetzten Einzelfiguren. Indem der Blick des Perzeptors der ästhetischen Information von Variante zu Variante wandert, erfaßt er durch Induktion die makroästhetische Innovation, die durch die Programmschleifen des Generators der Variation dispergiert worden ist. Eine radikal andere Perzeptivität erfordert die Erfassung der Textur. Im Stufenbau der ästhetischen Information nehmen die Texturen eine Mittelstellung ein, die von größter Bedeutung für die moderne Ästhetik ist. Funktional aufgefaßt gibt es ohne Textur keine Gestalt. Baumrinde, Ackerkrume, Sandboden, Tierfell, Holzmaserung, Rasenflächen, Getreidefelder, der Sternhimmel, die Erdoberfläche aus einer bestimmten Höhe gesehen, Lockengewirr, die Oberfläche vieler Gewebe, die Oberfläche fast aller Körper im Mikroskop betrachtet, allgemein ausgedrückt alles, was ohne Randkontur ist, oder aber innerhalb einer Randkontur keine Randkonturen mehr zeigt, sondern sich als ein Zeichenkontinuum bis an einen unscharfen oder scharfen Rand erstreckt — dieses alles ist Textur.

Texturen sind ästhetische Funktionen von Mikroinnovation und als solche sind sie die tragenden Substrate aller Makroästhetika, vor allem der Gestalten. *Substrat* selbstverständlich nicht naiv form-stoff-ontologisch aufgefaßt, sondern funktional.

Es gehört zum Wesen der Informationsästhetik, daß sie die Texturen zu den Ästhetika rechnet und besonders studiert. Dabei tritt das Textur-Substrat aus seiner reinen Trägerfunktion heraus und in objektivierende Perzeption ein. Objektivation von Textur ist aber dadurch ausgezeichnet, daß sie der gewöhnlichen Faulheit der

Perzeption, die beispielsweise ohne Umschweife den Begriff *Apfel* generiert, wenn ein Apfel perzipiert wird, ein radikales Ende setzt. Denn die gewohnte, makroperzeptorische Zuordnung von Objektnamen zu Strukturen setzt aus, wenn man Texturen untersucht. Allerdings ist gewohnheitsmäßige Namensassoziation auch bei Texturen nicht ausgeschlossen: *Samt* oder *Rauhputz* bilden Beispiele. Im allgemeinen jedoch erfordert das Studium von Texturen eine veränderte Betrachtungsweise. Wir bewegen uns jetzt im Bereich der Mikroästhetik und in AE, Seite 142, heißt es hierzu: „In dem Maße wie, nach einer Formulierung Hans Reichenbachs, in den mikrophysikalischen Theorien der Atome und Quanten die *Aussagen* über die Gegenstände (also *physikalische Sprachen*) an die Stelle von *physikalischen Gegenständen* treten, die das Thema der Makrophysik bilden, treten auch in der Mikroästhetik ästhetische Zeichen (Rhythmus, Metrum, Farb-Form-Verhältnisse, syntaktische Partikel, Bedeutungen, Worte selbst, Farben selbst) an den Platz der dargestellten Gegenstände (wirkliche Dinge, Szenen, Fabeln, Handlungen, Konflikte usw.), die der makroästhetischen Welt angehören."

Kann die generative Graphik zur Aufklärung des Strukturproblems der Texturen beitragen? Bislang benutzten wir lediglich einen naiven, exemplarisch gewonnenen Texturbegriff, der informationsästhetisch zu verschärfen ist. Dies wollen wir jetzt versuchen. In jedem Fall eignet der ästhetischen Information, die durch Texturen ausgebreitet und dargeboten wird, eine gewisse Monotonie. Voneinander entfernte, nicht unmittelbar benachbarte Orte der Textur haben nichts miteinander zu tun, keine makroästhetischen Gebilde verbinden sie. Es existieren keine weitreichenden Spannungsbögen, durch die das gleichförmige Einerlei der lokalen Gegebenheiten der

Textur in selbständige Informationseinheiten zerlegt würde. Texturen sind also keinesfalls Gestalten, vielmehr sind sie deren Antagonisten. Haben die Texturen etwas mit den Haufen zu tun, die laut Figur 7 ebenfalls zu den Antagonisten der Gestalten zählen? Der Unterschied zwischen Haufen und Texturen kann vielleicht folgendermaßen erfaßt werden: In Haufen springt man perzeptorisch von Informationselement zu Informationselement, dagegen schreitet man in Texturen wie in einförmigen Wüsteneien vorwärts, von wesentlichen Hindernissen nicht gestört. Texturen sind also Informationskontinua. Was stiftet den inneren Zusammenhang der Textur? Makroästhetische Klammern existieren nicht. Texturen sind also Informationskontinua mit Lokalnexus ohne Distalnexus. Im Gegensatz dazu sind Haufen Diskontinua mit Distalnexus ohne Lokalnexus (fehlte jeder Nexus, so lieferten die Haufen keinerlei ästhetische Nachricht, keine Innovation, was sie tatsächlich tun). Wir können jetzt sogar versuchen, eine informationsästhetische Definition der Gestalt anzuschließen: Gestalten sind ästhetische Informationseinheiten mit Lokal- und Distalnexus.

Noch haben wir einen der wichtigsten Begriffe der Informationsästhetik nicht in die Diskussion gebracht: Nichts ist bis jetzt über den Zusammenhang zwischen Mikroinnovation und Textur ausgesagt worden. Wir wollen diese Problematik jedoch zugunsten praktischen Experimentierens mit dem Texturbegriff zurückstellen und uns zunächst fragen, ob und wie die generative Graphik mit Hilfe des Instrumentariums, mit dem wir bereits vertraut sind, möglichst verschiedenartige Texturen — Modelltexturen sozusagen — erzeugen kann. Der Umgang mit Texturen wird uns die Weiterverfolgung ihrer Theorie erleichtern.

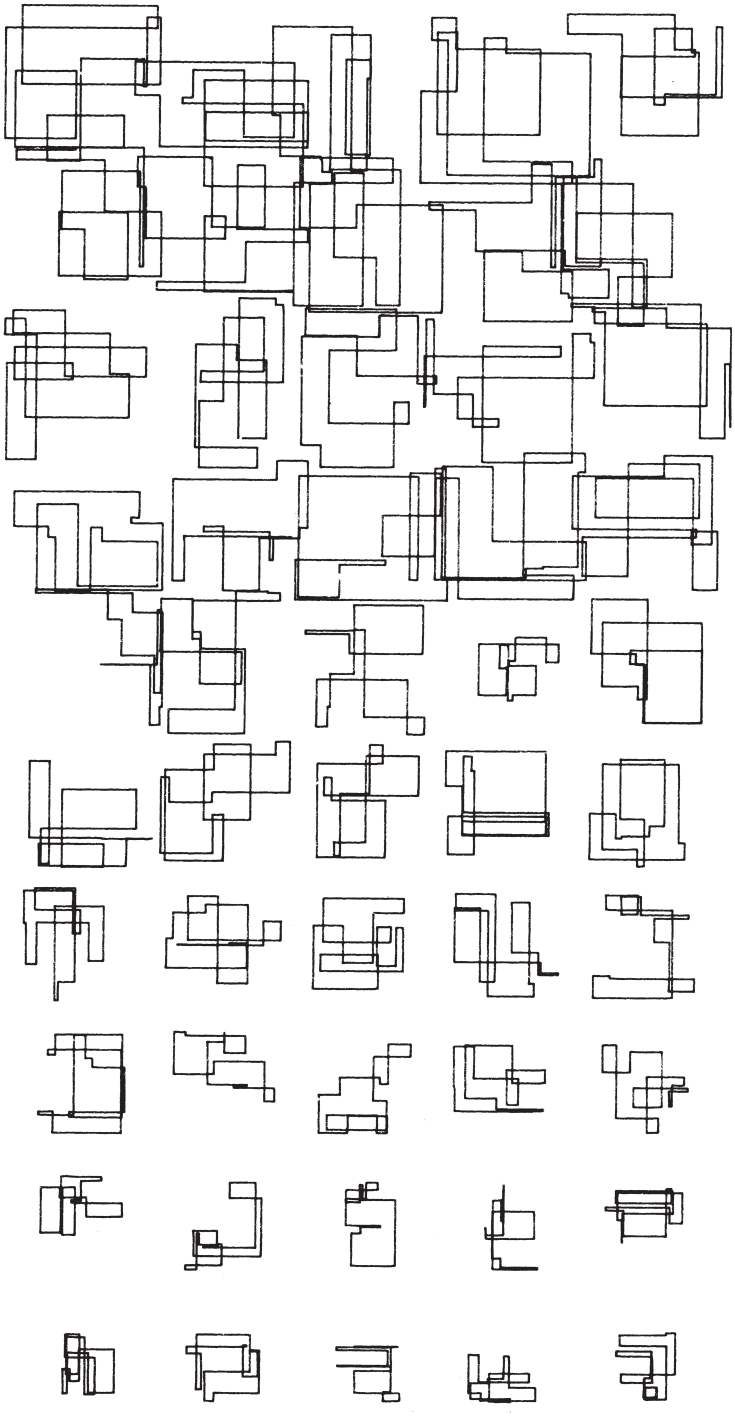
Bei der Erzeugung von Texturen sind der generativen Graphik Grenzen gesetzt. Eines der auffälligsten Merkmale der Klasse der Texturen ist ja, daß so viele ihrer Vertreter nicht durch Aufbau-, sondern durch Abbau-, Zerstörungs- und Einebnungsprozesse zustandekommen: Die Verwitterung schafft Texturen — Gesteinsoberflächen und Wüstenlandschaften. Sehr glatte Metallflächen, die als Ästhetika am äußersten Rand des Bereichs der Texturen stehen, verwandeln sich durch Korrosion in echte Texturen. Korrosionsprozesse können wir innerhalb der generativen Graphik höchstens dadurch nachahmen, daß wir zufällige Marken so auf die Zeichenfläche setzen, daß sie nach Art von Korrosionsmarken in ihrer Gesamtheit die ursprüngliche Struktur der Zeichenfläche ersetzen. Bild 33 zeigt das Ergebnis eines solchen Versuchs. Auch Texturgenese durch schrittweisen Zerfall von Anstückungen gleichartiger Blöcke — der Fußboden, nach dem Spiel der Kinder mit Bauklötzchen bedeckt — können wir nur simulativ erfassen; wir werden später eine derartige Textur untersuchen. Es gibt jedoch Verfahren der Texturgenese, die der generativen Graphik sehr leicht fallen, die aber trotzdem Einblick in die Texturproblematik gewähren. Ein solches Verfahren werden wir jetzt erörtern: Wir werden Texturen generieren, indem wir die Varianten von Variationen miteinander verhaken oder verflechten. Was wir dabei erzeugen, ist natürlich Lokalnexus — der jedoch ist das Hauptkennzeichen des ästhetischen Typs der Textur. Übrigens enthält auch diese Methode einen stark destruktiven Zug insofern, als ja durch die gegenseitige Überlappung die Individualität der Einzelfiguren der Variation zerstört wird. So, wie der Überlappungseffekt den Schwarmcharakter einer Bildinformation tilgen kann (siehe Abschnitt 3.1.3), eliminiert er jetzt den ästhetischen Modus der Figurenreihung.

Die Verflechtung von Variationen erreicht man, indem man die Einzelfiguren nicht mehr räumlich voneinander absetzt, sondern gegenseitig übereinandergreifen läßt. Das ist jedoch mit der Maschinerie von SERIE und ELIRR sehr leicht zu bewerkstelligen. Man braucht ja nur die in Figur 5 Abschnitt 3.2.2 dargestellten Größen JLI, JRE, JUN, JOB in der Vereinbarung von ELIRR geeignet abzuändern (siehe (1/3.2.2)). Z. B. erzeugt der folgende Block die Texturen der Bilder 29 und 30, wenn man ihn genauso in das Gesamtprogramm einbettet wie den von Zeile 6 bis Zeile 17 sich erstreckenden Block von (1/3.2.3):

```
(1)   6  'BEGIN' 'COMMENT' TEXTUR
      7  AUS IRRWEGRECHTECKEN
      8  0 UND 1.,
      9  JI1.=JS2., JI2.=JS5.,
     10  N.=20.,
     11  JLI.=−10., JRE.=30.,
     12  JUN.=−10., JOB.=30.,
     13  T.=0.,
     14  SERIE(20.0,20.0,8,6,ELIRR).,
     15  T.=1.,
     16  SERIE(20.0,20.0,8,6,ELIRR).,
     17  'END' TEXTUR
     18  AUS IRRWEGRECHTECKEN 0 UND 1.,
```

Die beiden erwähnten Blöcke unterscheiden sich wesentlich überhaupt nur in den Zeilen 11 und 12. Das Elementarrechteck, in das SERIE die Zeichenfläche einteilt, hat ebenso wie bei den Bildern 22 und 23 die Abmessungen 20 mal 20 Längeneinheiten, wie aus den Prozeduraufrufen in den Zeilen 14 und 16 von (1) hervorgeht. Das Rahmenrechteck, in das ELIRR den elementaren Irrweg einbeschreibt, greift jetzt jedoch links, rechts, oben und

Bild 28



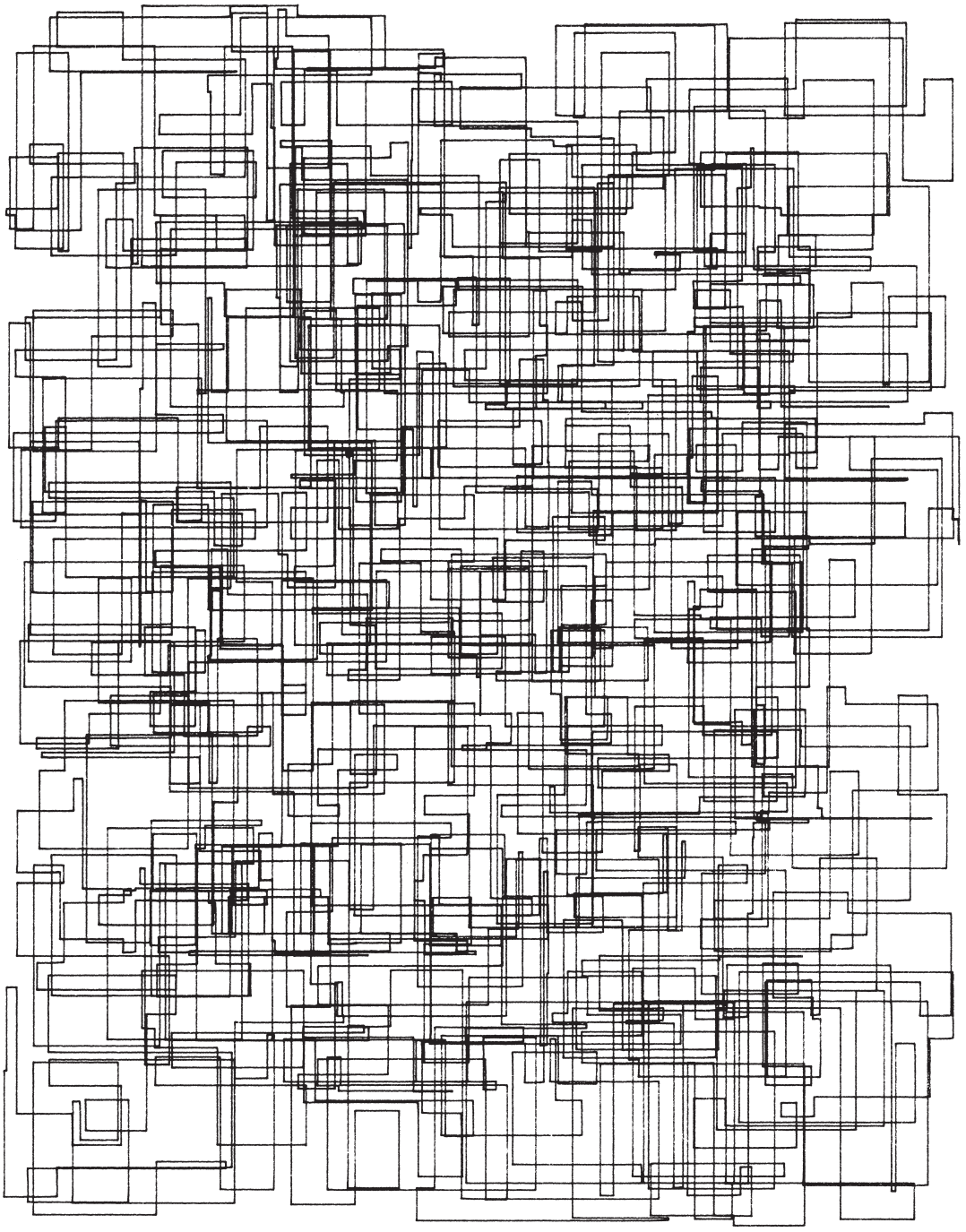
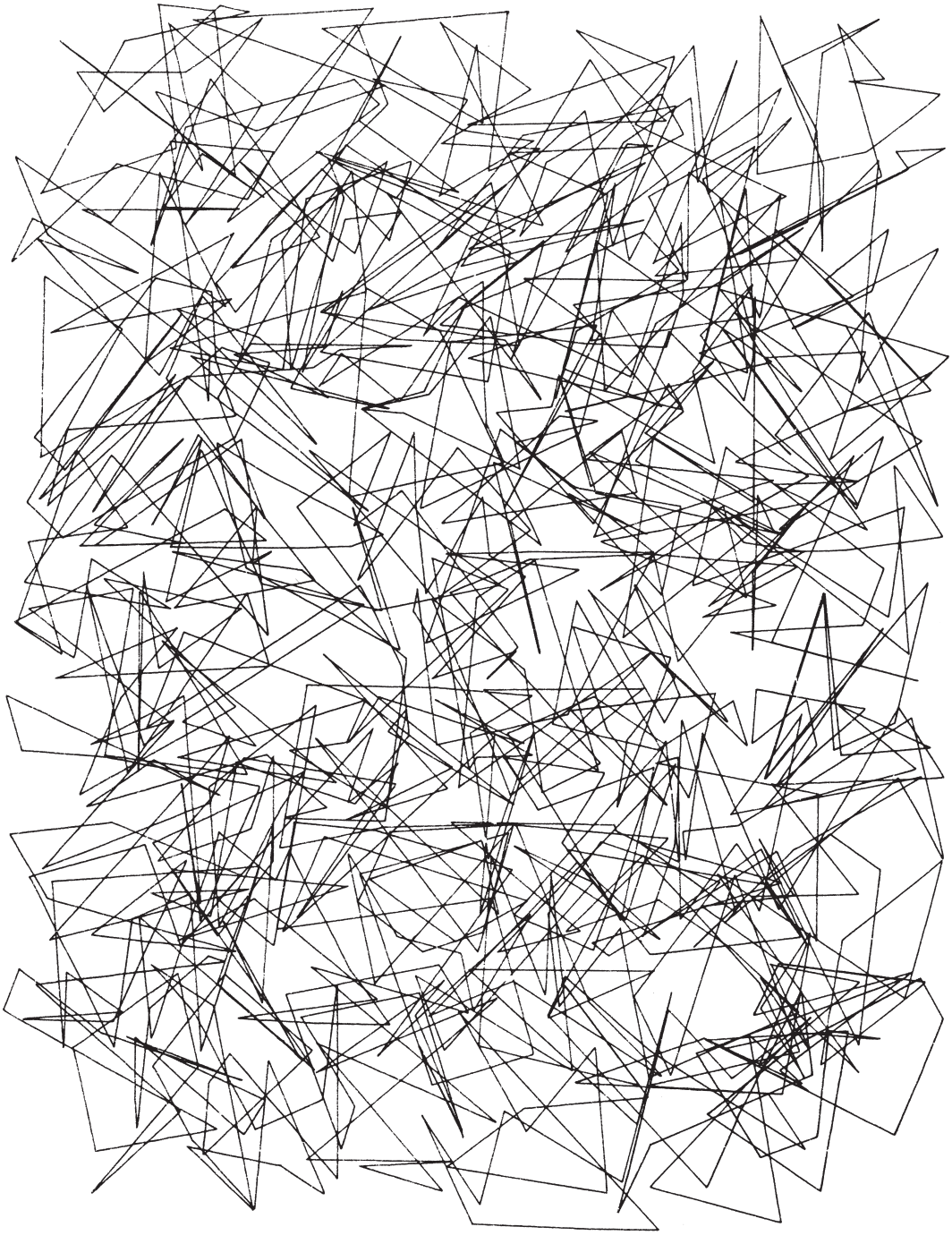


Bild 29

Bild 30



unten um 10 Längeneinheiten über das Elementarrechteck von SERIE hinaus, so daß die beiden Rechtecke genauso gegenseitig gelagert sind, wie es in Figur 6 dargestellt wurde (siehe Abschnitt 3.2.4). Die Folge ist die gewünschte gegenseitige Überschiebung der Einzelfiguren der Variation. Man kann durch einfache Betrachtung der Bilder 22 und 29 noch die Korrespondenz von Einzelfiguren herstellen: Im Hochformat betrachtet, fällt z. B. ein sich in den linken unteren Ecken der Bilder nach oben erstreckender Haken auf, der lediglich in Bild 29 wegen der Vergrößerung des Rahmenrechtecks von ELIRR größer ist als in Bild 22.

Gewissermaßen dynamisch erfaßt man den Übergang vom Variationenbild zur Textur, wenn man den Überlappungseffekt im Verlauf einer Durchvariierung schrittweise steigert. Bild 28 zeigt das Ergebnis eines solchen Versuchs. Wesentliches Hilfsmittel ist eine Prozedur ELIRRW, die der Prozedur ELIRR übergestülpt wird und die von sich aus eine Ausdehnung des Rahmenrechtecks für den Irrweg jedesmal bewirkt, wenn sie aufgerufen wird. Man findet ELIRRW in den Zeilen 4 bis 8 des Generators (2) für Bild 28.

(2)

```
 1 'BEGIN'
 2 'COMMENT'
   UEBERGANG VARIATION ZU TEXTUR.,
 3 'REAL' V.,
 4 'PROCEDURE' ELIRRW.,
 5 'BEGIN' ELIRR.,
 6 JLI.=JLI-V., JRE.=JRE+V.,
 7 JUN.=JUN-V., JOB.=JOB+V.,
 8 'END' ELIRRW.,
 9 JI1.=JS1., JI2.=JS2.,
10 N.=10., T.=0., V.=.2.,
```

```
11  JLI.=5.,JRE.=15.,
12  JUN.=5.,JOB.=15.,
13  SERIE(20.0,20.0,10,5,ELIRRW)
14  'END'
    UEBERGANG VARIATION ZU TEXTUR.,
```

Texturerzeugung durch Überlappung von Einzelfiguren ist ein sehr ergiebiges Verfahren. Da wir bereits im Besitz des Transduktors SERIE sind, ist es überdies leicht zu handhaben. Durch die Möglichkeit, den Parameter FIGUR von SERIE und die gegenseitige Lage der durch FIGUR erzeugten Figur zum Elementarrechteck von SERIE frei zu wählen, ergibt sich eine vorerst unüberschaubare Mannigfaltigkeit von Texturen. Bild 34 greift ein Beispiel heraus. Dieses „Mikadospielhaufen I" genannte Bild entsteht durch Ersetzung von FIGUR durch eine Prozedur STAB, die Strecken konstanter Länge, jedoch variabler Schräglage erzeugt. Der Generator für den Mikadospielhaufen I sieht folgendermaßen aus:

(3)

```
 1  'BEGIN' 'COMMENT'
    MIKADOSPIELHAUFEN I.,
 2  'REAL' PSI, PI.,
 3  'PROCEDURE' STAB.,
 4  'BEGIN' PSI.=J1.,
 5  LEER(P+2.5+30*COS(PSI),
    Q+2.5+30*SIN(PSI)).,
 6  LINE(P+2.5+45*COS(PSI+PI),
    Q+2.5+45*SIN(PSI+PI))
 7  'END'.,
 8  PI.=3.14159.,JI1.=JS1.,
 9  JA1.=0.,JE1.=PI.,
10  SERIE(5.0,5.0,26,18,STAB)
11  'END' MIKADOSPIELHAUFEN I.,
```

Die Doppelzeilen 5 und 6 lassen erkennen, daß die Prozedur STAB eine Strecke von 15 Längeneinheiten im Winkel PSI zur X-Achse zeichnet. Der Wert von PSI wird in Zeile 4 durch den Zufallsgenerator J1 bestimmt, dem in Zeile 9 Streugrenzen zwischen 0 und 180 Grad gesetzt werden. Man erkennt, daß man die Makroinnovation in Bild 34 durch Einengung der Streugrenzen von J1 steigern könnte.

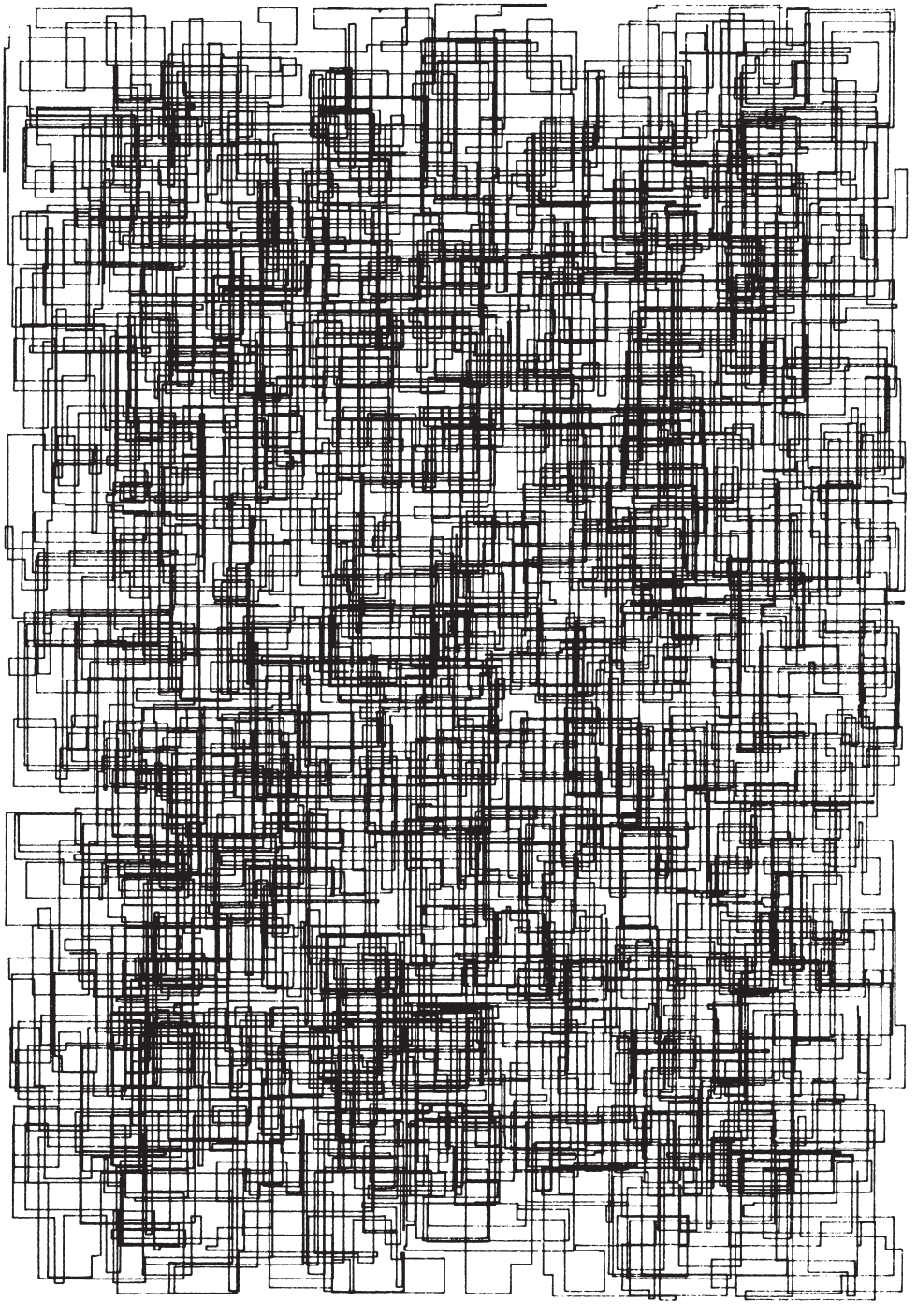
Wie die Bilder 29 und 30 zeigen, führt schon die Einsetzung der uns wohlbekannten Prozedur ELIRR für FIGUR zu brauchbaren Texturbeispielen. Dieser Eindruck verstärkt sich bei Betrachtung der Bilder 31 bis 33, deren Generatoren sich nur in den Elementar- und Rahmenrechtecken für SERIE und ELIRR und durch die jeweils verschiedenen Anzahlen der Elementarrechtecke und Irrwegkanten unterscheiden. Wir lassen nun den Generator für diese drei Bilder folgen:

(4)

```
1  'BEGIN' 'COMMENT'
   TEST MIKROINNOVATION.,
2  T.=0., JI1.=JS1., JI2.=JS5.,
3  JLI.=JUN.=−8., JRE.=JOB.=24., N.=32.,
4  SERIE(16.0,16.0,12,8,ELIRR).,
5  JLI.=JUN.=−4., JRE.=JOB.=12., N.=8.,
6  SERIE(8.0,8.0,24,16,ELIRR).,
7  JLI.=JUN.=−2., JRE.=JOB.=6., N.=2.,
8  SERIE(4.0,4.0,48,32,ELIRR)
9  'END' TEST MIKROINNOVATION.,
```

Man erkennt den völlig uniformen Aufbau der Zeilenpaare 3, 4 usw., die zur Erzeugung der Bilder 31 bis 33 dienen. Das Elementarrechteck ist zum Rahmenrechteck wie in Figur 6 gelagert, jedoch halbieren sich die Kantenlängen beider Rechtecke von Bild zu Bild. Bei Verwen-

Bild 31



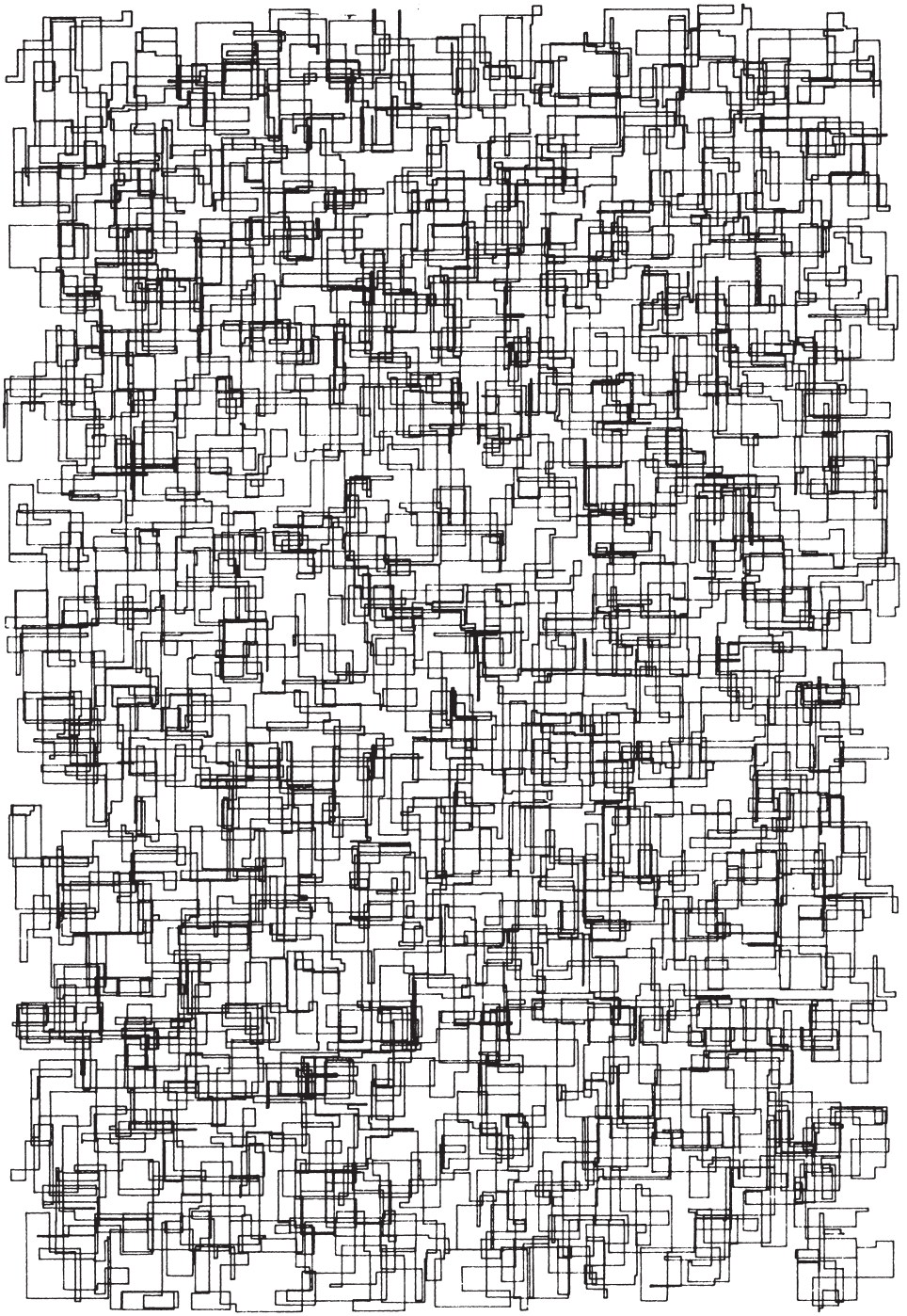


Bild 32

Bild 33

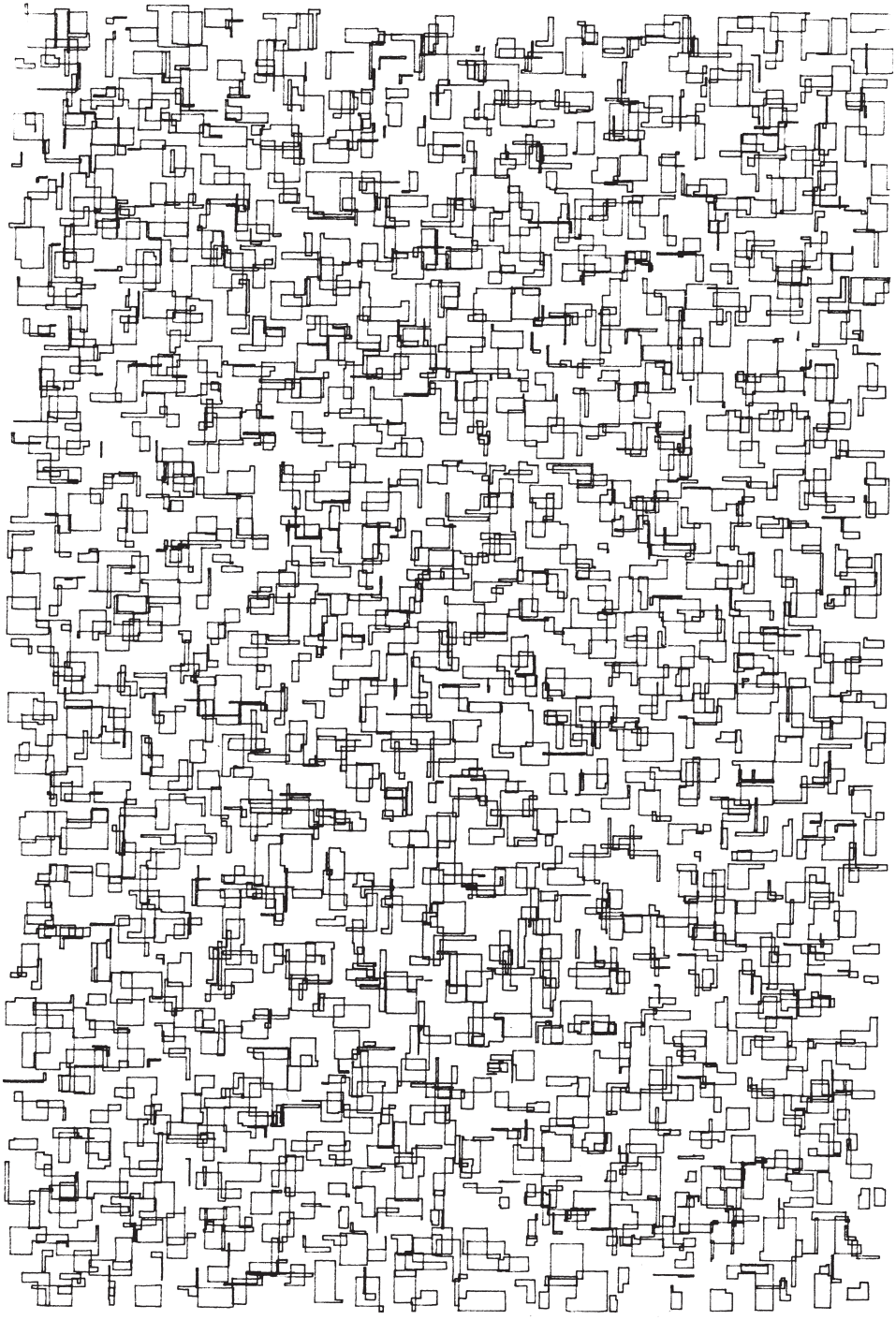
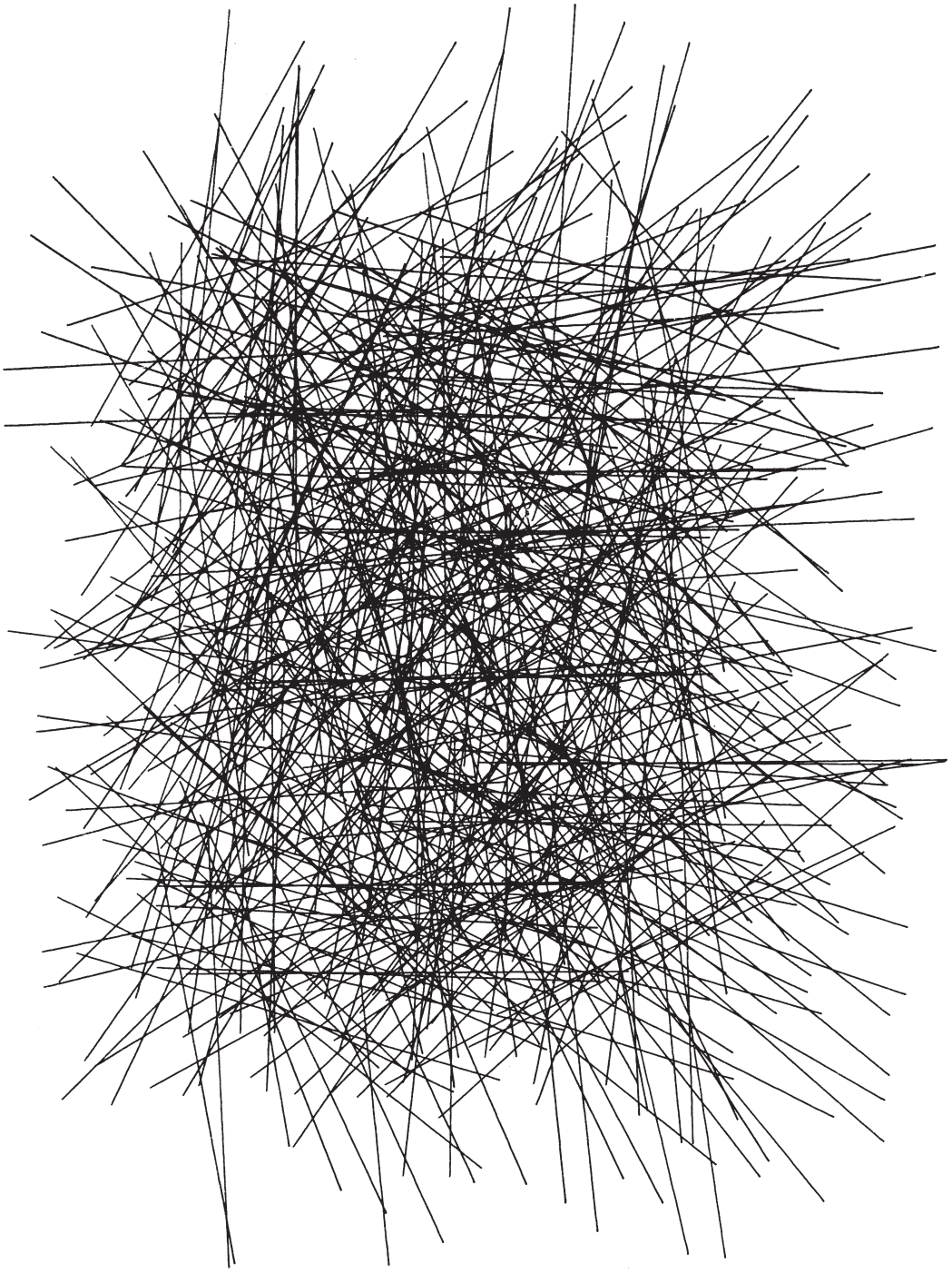


Bild 34



dung von ELIRR gehorchen die Eckpunktanzahlen des elementaren Irrwegs der Formel $K=2*(N+1)$, haben also in unseren Beispielen die Werte 66, 18 und 6. Damit sinken die Eckpunktanzahlen von Bild 31 bis Bild 33 etwas schwächer ab, als die zweiten Potenzen der Kantenlängen der Elementarrechtecke. Aus dieser Zahlenkabbalistik folgt leider nicht ohne weiteres Brauchbares über die qualitativen Merkmale der Bilderserie; zwar stellt sich von Bild zu Bild eine Art stetiger Übergang des Charakters der ästhetischen Information ein, jedoch ist Bild 31 von Bild 33 qualitativ völlig verschieden. Es gilt, anders ausgedrückt, kein transitives Gesetz der Ähnlichkeit (wenn man vom Merkmal der Kantenparallelität absieht). Das Beispiel dieser Serie läßt ahnen, wie schwer es eine künftige quantitative Theorie der Texturen haben wird.

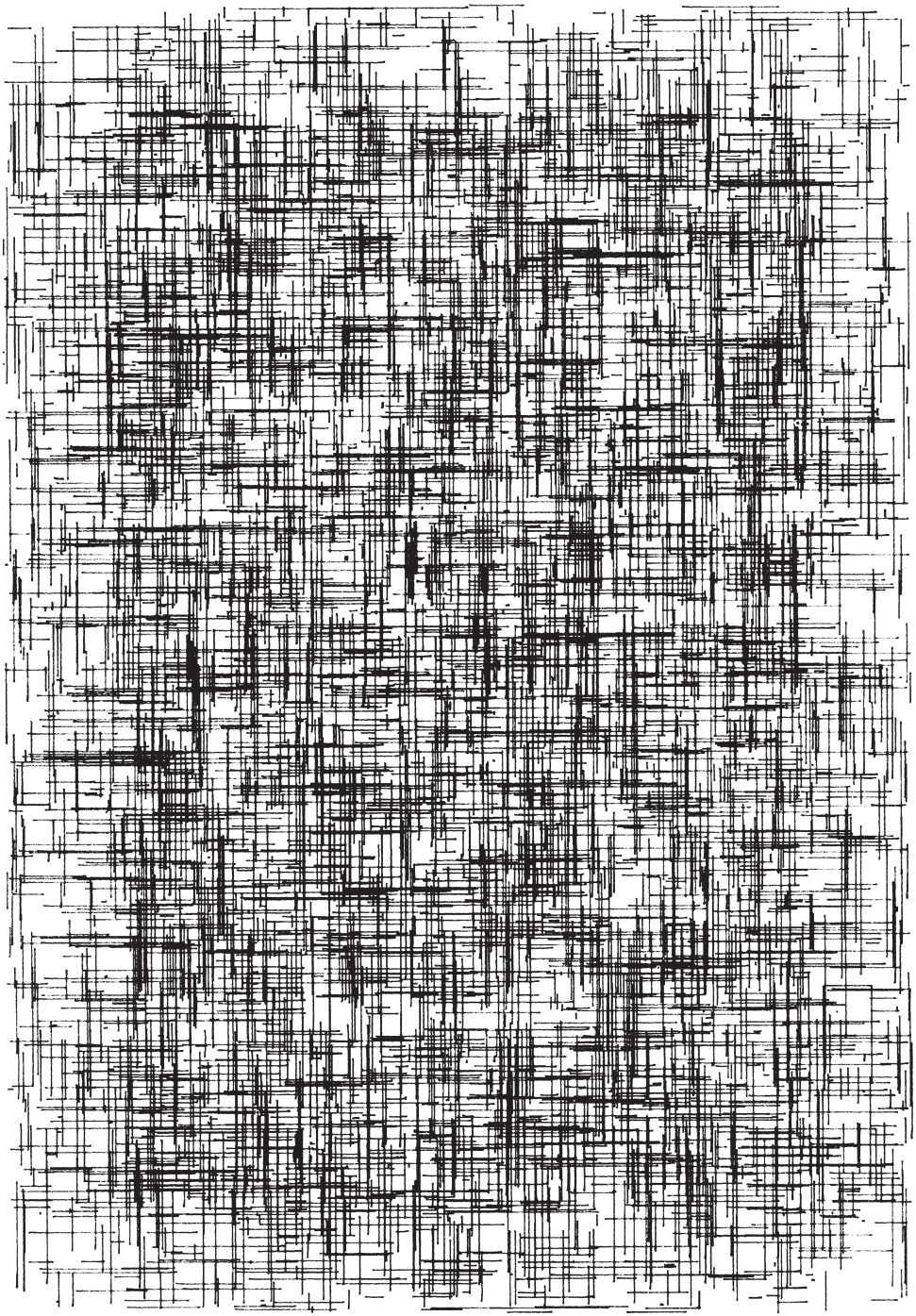
Hier ist wohl die richtige Stelle, sich einer Grundfrage der generativen Graphik zu entsinnen, die wir am Schluß der Einführung zu Abschnitt 3.1 stellten: *Was sind die Anteile von Innovation und Redundanz an der Zeichenkonstellation, die durch einen bestimmten Generator erzeugt wird?* Am Schluß von Abschnitt 3.1.5 sagten wir, daß die Bildredundanz durch Dispersion aus einem Modus von Innovation hervorgeht. Da die Redundanz in der erfundenen Struktur des Generators gründet, enthält sie keine mikroinnovativen Anteile, vielmehr ist sie rein induziert. Globale, makroästhetische Effekte der Bildinformation determinierend, stammt sie von Makroinnovation her. Im Fall der Textur ist nun das entscheidend Wichtige, daß die induzierte Makroinnovation einen sehr geringen Einfluß ausübt. Z. B. ist bei den durch Überlappung erzeugten Texturen die typische induzierte Makroinnovation auch schon vor der Überlappung da, wie Bild 28 beweist. Es müssen also andere Innovationsmo-

dalitäten das Übergewicht im Bildcharakter der Textur bestimmen.

Wie wir wissen, ist neben der Redundanz oder induzierten Makroinnovation, die im Weichen- und Schleifenaufbau des Bildgenerators gründet, die von den Zufallsgeneratoren verursachte Mikroinnovation die zweite fundamentale Komponente der Bildinformation. Ist nun der Einfluß der induzierten Makroinnovation auf die Struktur der Textur schwach, so ist der Einfluß der Mikroinnovation um so stärker. Texturen sind wesentlich Mikroinnovation, wobei sie der Rest an induzierter Makroinnovation, den sie an sich haben, vor dem Versinken im unerkennbaren Chaos rettet.

Die mikroinnovative Komponente der Textur ist deshalb so übermächtig, weil sie durch die mit der Überlappung einhergehenden lokalen Verhakungen, Verknüpfungen, Verschweißungen von Bildelementen den perzeptorisch in jedem ihrer Teile uniform erfaßten Charakter der Textur festlegt. Der lokale, jedoch überall in der Textur gleichartige Bildnexus, der sich dadurch herstellt, ist das Spezifische der jeweiligen Textur. Ein kleiner Ausschnitt aus Bild 31 z. B. ist so, wie er perzeptorisch erfaßt wird, nicht schon ableitbar aus dem Wissen, daß sich hier waagrechte und senkrechte Linien überkreuzen. Den Beweis für diese Behauptung liefert Bild 35, für die sie ebenfalls zutrifft (wir schreiben den extrem einfachen Generator für Bild 35 nicht an). Makroästhetisches spielt natürlich eine Rolle im Aufbau der Textur, das wird ja beispielsweise durch den Unterschied zwischen den Bildern 31 und 33 bewiesen. Aber wie es hereinspielt, kann aus dem Aufbau und den Parameterwerten des Texturgenerators nicht vorhergesagt werden. Die generative Textur ist tückisch, sie bricht aus dem Schema aus, das

Bild 35



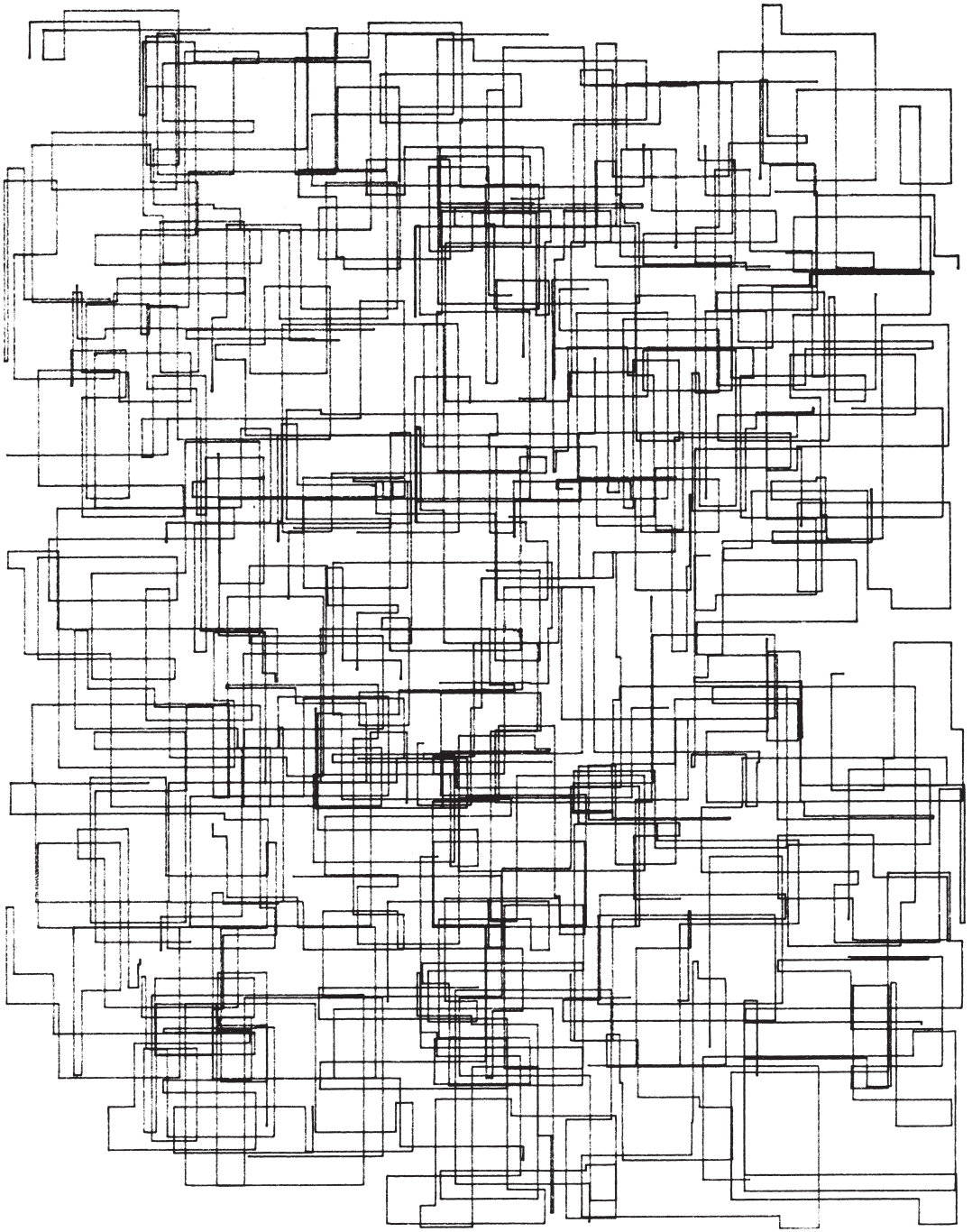
ihr der Generator zu setzen versucht. Noch genauer kann man die Funktion der Mikroinnovation im Gefüge der Textur erfassen, wenn man es erreicht, Aussagen über die verschiedene örtliche Beschaffenheit verschiedener Texturen zu machen. Schon weiter oben sagten wir, daß sich die Textur wesentlich durch ihre örtliche Beschaffenheit darstellt, da sie keine räumlich weit ausgreifenden Bildkomponenten enthält — dabei sehen wir natürlich von Rändern und deren Wirkungen ab, die durch das Format des Zeichenblatts erzwungen werden. Wir nannten die örtliche Beschaffenheit der Textur ihren Lokalnexus. Im nächsten Abschnitt werden wir uns an einer einfachen Theorie des Lokalnexus versuchen.

3.2.6.
Lokale Innovationsdichte Endogene Redundanz

Um den Begriff der lokalen Innovationsdichte definieren zu können, kommen wir auf die Erkenntnis zurück, daß die Bildinformationen Zeichenkonstellationen sind, Systeme von Zeichen also. Innerhalb des Bildes als eines Gesamtzeichens kann man Elemente entdecken, die das Gesamtzeichen kompositiv aufbauen. Auch im Lokalnexus der Textur ist Zeichenhaftes erkennbar, das den Lokalnexus als solchen bestimmt. Diese zeichenartigen Elemente hat Max Bense mit dem Ausdruck *syntaktische Partikel* belehnt (siehe die Stelle in AE, Seite 142, die wir schon im vorhergehenden Abschnitt zitierten). Zweifellos sind die syntaktischen Partikel selbst Zeichen. Da sie jedoch im Lokalnexus der Textur nur sehr schwer isoliert werden können, tatsächlich eher indirekt, wenn auch determinativ, in den Gesamteindruck des Lokalnexus eingehen, nennen wir sie Protozeichen. Als lokale Innovationsdichte einer Textur bezeichnen wir nun die Lagerungsdichte der Protozeichen im Lokalnexus der Textur.

Wir sind uns bewußt, daß unsere Definition der lokalen Innovationsdichte der Exaktheit entbehrt. Um ihre Brauchbarkeit zu erweisen, holen wir noch etwas weiter aus. Wenn man den Lokalnexus einer bestimmten Textur genauer zu kennzeichnen versucht, so stößt man auf eigentümliche und durchaus tiefliegende Schwierigkeiten. Es stellt sich nämlich schnell heraus, daß die Umgangssprache für die vollständige Erfassung des Lokalnexus nicht ausreicht. Die Unzulänglichkeit der sprachlichen Beschreibung ist hier so gemeint, daß sie nur selten gestattet, im potentiellen Perzipienten einen brauchbaren Begriff der beschriebenen Textur wachzurufen. Im Bereich der gegenständlichen Bedeutung ist das ganz anders: Die Nennung von *Rose* oder *Nelke* unterrichtet den Vernehmenden dieser Klassennamen sowohl bildlich-anschaulich, als auch begrifflich genau über das jeweils gemeinte. Bei Texturen gelingt solche Informierung nur dann, wenn sie gebrauchsmäßig bekannt sind, wie z. B. die im vorhergehenden Abschnitt erwähnten Texturen von Samt und Rauhputz. Betrachten wir beispielsweise Bild 29, so können wir wohl anfangen den Lokalnexus zu beschreiben und sagen: *Rechtwinklig Haken schlagende, achsenparallele Linien überlagern sich mit mittlerer Dichte.* Wer aber vermöchte sich, allein aufgrund dieser sprachlichen Information, ein klares Bild von der intendierten Textur zu machen, wenn er sie noch nicht gesehen hat? Bild 36, das man aus Bild 29 dadurch erhält, daß man den Irrweg von ELIRR nicht schließt, ist mit den gleichen Worten beschreibbar; eine gewisse Differenzierung Bild 29 gegenüber ist dadurch herstellbar, daß man die Angabe hinzufügt: *Die rechtwinklig Haken schlagenden Linien beginnen und brechen bald wieder ab.* Das alles bleibt jedoch blaß gegenüber der optisch perzipierenden Erfassung der Textur. Bei der Untersuchung der Texturen schält sich ganz

Bild 36



unzweifelhaft heraus, daß die Rezeptionspotenz des Sehrezeptakulums als eines Ganzen wesentlich größer ist als die Ausdrucksfähigkeit der informationsdiskretisierenden Umgangssprache.

Im Grunde war schon das gleiche Phänomen gemeint, als wir im vorhergehenden Abschnitt von Bild 31 und Bild 35 sagten, daß beide Bilder sich überkreuzende waagrechte und senkrechte Linien darstellten. Wir machten dort die Verknüpfung von Zeichenelementen durch die Überlappungswirkung für die Mehrdeutigkeit der letzten Endes makroästhetisch orientierten Bildbeschreibung verantwortlich. Tatsächlich sind die Überlappungseffekte sehr weitgehend mikroinnovativ bestimmt, so daß wir jetzt den Zusammenhang zwischen Textur und Umgangssprache so formulieren können: *Die Unzulänglichkeit der Texturbeschreibung ist eine Folge des Übergewichts der Mikroinnovation im Texturaufbau.* Dadurch wird die Texturforschung zu einer schwierigen Angelegenheit, auf der anderen Seite liegt es freilich in der funktionellen Natur der Textur, daß sie subsprachlich ist. Andernfalls könnte sie wohl nicht als Substrat makroästhetischer, bedeutungsgeladener Zeichen fungieren. Sie würde dauernd ihre eigene Begrifflichkeit in die von ihr getragenen Bedeutungen einschieben.

Die Unmöglichkeit vereindeutigender Texturbeschreibung vergrößert den Wert genereller Texturkennzeichnungen, also auch den der lokalen Innovationsdichte. Auch der Begriff des Protozeichens wird jetzt deutlicher. Zu den Protozeichen haben wir nämlich auch Zeichen im Lokalnexus zu zählen, die ihre Existenz allein dem Überlappungseffekt verdanken. Protozeichen sind in Bild 29 Linienverdickungen und kleine, rechtwinklig begrenzte Flächenbezirke. In Bild 30 sind sternartige Linienzer-

splitterungen, in Bild 34 ebenfalls Liniensterne und winzige Polygone Protozeichen. Bild 33 ist ein ausgesprochener Grenzfall, insofern, als hier der Überlappungseffekt nur schwach wirksam ist und sich Protozeichen als echte elementare Bildkomponenten herausstellen. Aus großer Nähe betrachtet, trifft sich Bild 33 mit dem Haufenbild 27. Die lokale Innovationsdichte verschiedener Texturen ist anschaulich vergleichbar. Bei unseren Beispielen ist sie vielleicht am geringsten in Bild 30. Qualitativ in eine Klasse gehören die Texturen 29, 31 und 32. Hier wächst die lokale Innovationsdichte mit der Bildreihenfolge 29, 32, 31. Das ist insofern paradox, als das Elementarrechteck bei Bild 32 nur halb so groß ist als bei Bild 31, man also bei Bild 32 die größere Lagerungsdichte der Protozeichen erwarten würde. Das Paradoxon zeigt wieder die Unmöglichkeit auf, beim Programmieren schon alle Wirkungen des Überlagerungseffekts vorauszusehen. Prägt man vorsichtig und versuchsweise einen Begriff *Gestaltetheit des Lokalnexus,* so würde diese Eigenschaft als von Bild 31 bis Bild 33 ansteigend zu behaupten sein. Bemerkenswert ist auch, daß sich die Innovationsdichten der Bilder 31 und 34 als ungefähr gleich hoch ansprechen lassen, obgleich ihre Lokalnexus vollständig verschieden sind.

Wir wissen also jetzt, daß der makroästhetische Bau des Generators nicht ohne weiteres auf die mikroinnovativen Eigenschaften der generierten Textur schließen läßt. Ganz grob läßt sich vielleicht behaupten, daß die lokale Innovationsdichte um so größer sein wird, je mehr und komplexere und ausgedehntere Zeichen der Generator bei der Texturgenese miteinander verknüpft. Noch allgemeiner ausgedrückt: Je stärker der Generator Überlappungseffekte begünstigt, desto dichter und gestalteter wird der Lokalnexus der Textur sein.

Die örtliche Beschaffenheit der Textur, die wir auch als ihren Lokalnexus bezeichnen, ist überall in der Textur die gleiche, wenn auch nicht translativ von einer Stelle an die andere identisch übertragbar. Wie wir gesehen haben, ist der Lokalnexus eine Funktion sowohl der induzierten Makroinnovation als auch der Mikroinnovation, wobei der Einfluß der Mikroinnovation um so mehr überwiegt, je stärker Überlappungseffekte ins Spiel kommen. Bei der kontinuierlichen Perzeption einer Textur wird die Eigenart der Textur in der Weise wahrgenommen, daß die Konstanz des Lokalnexus vom Perzipienten als Bildredundanz aufgefaßt wird. Früher nun bezeichneten wir die Bildredundanz als rein makroästhetische Angelegenheit. Das Originelle an der Bildredundanz ist ja auch im Normalfall Makroinnovation. Dieser theoretische Standpunkt läßt sich nicht in voller Allgemeinheit aufrechterhalten, wenn Überlappungseffekte die Bildinformation wesentlich mitbestimmen. Dann ist nämlich die Redundanz, wie der Lokalnexus, eine Funktion von Makroinnovation und Mikroinnovation. Spaltet man wenigstens methodologisch die Bildredundanz in einen makro- und einen mikroinnovativ bestimmten Anteil auf, so kann der zweite Anteil als inhärent bezeichnet werden. Schon in Abschnitt 3.1.6 unterschieden wir induzierte und inhärente Makroinnovation. Anstatt *inhärent* kann man mit gutem Recht auch *endogen* sagen. Da Distalzusammenhänge im Gefüge der Textur nicht vorkommen, gibt es in ihr keine endogene Makroinnovation, wohl aber endogene Redundanz. Makroinnovative Redundanzanteile können den endogenen gegenüber als induziert oder auch als exogen gelten, wenn wir unseren Begriffsvorrat so ergänzen wollen. Vom genetischen Standpunkt aus gibt es also Exogeneität und Endogeneität beim Aufbau der Textur. Exogenes kommt vom Programmierer, es ist induziert. Endogenes

kommt oft unvorhersehbar aus dem Innern der Graphik, es ist inhärent.

Die lokale Innovationsdichte graduiert alle Texturen, reiht sie in eine Gradation ein. Die äußersten Enden dieser Gradation sind ziemlich unbestimmt. Texturen mit sehr geringer lokaler Innovationsdichte schlagen in Haufenbilder oder in heterogene Ansammlungen von einzeln bedeutungsvollen Figuren um, ein Beispiel bildet der Falterschwarm in Bild 4. Steigt die Innovationsdichte sehr stark an, so übersteigt das Kompaktum von gestalteten Einzelheiten im Lokalnexus schließlich die informationsanalysierende Kraft des Perzipienten, die Textur nähert sich dem Chaos. Eine derartige Textur erhielte man durch unbegrenzte Fortführung des Irrwegs von Bild 12 innerhalb der eingezeichneten Begrenzung. Im mittleren Bereich liegen die Texturen, die der Perzipient normalerweise antrifft, und übersieht, wenn sie in reiner Trägerfunktion auftreten. Im mittleren Bereich hat die Textur eine Redundanz, d. h. einen Grad der Wiederholung von Innovativem, der eine gewisse Angepaßtheit und Gewöhnung des Perzipienten entspricht. In Fortführung dieser Überlegung kann man die Texturen mittlerer Redundanz als praktikable Entmischungen aus dem Chaos auffassen. Durch diese Entmischung realisieren sich die Texturen und werden fähig, in die gestalttragende Substratfunktion einzutreten. Max Bense faßt das Chaos als Potentielles und zitiert hierzu eine Stelle aus der Metaphysik des Aristoteles, an der dieser den Begriff des Potentiellen mit den Begriffen des chaotischen Durcheinander bei Anaxagoras, der Mischung bei Empedokles und Anaximander und der Bemerkung *Es war alles durcheinander* bei Demokrit in Verbindung bringt. Bense fährt fort (AE, Seite 229): „Interessant an dieser Stelle sind zwei Dinge, erstens, daß Realisation

(wie in der Informationstheorie WIENERS) soviel bedeutet wie Übergang von Unordnung zu Ordnung, von Mischung zu Entmischung und zweitens, daß sich darin der Übergang vom Möglichen zum Wirklichen vollzieht, das Mögliche, das Potentielle also offenbar als ungeordneter Zustand der Mischung aufgefaßt wird." Man könnte Bild 28 in diesem Sinn deuten: Es kann als Texturentmischung bis zu einem Grad aufgefaßt werden, bei dem der Texturbegriff verlorengeht und Gestalten erscheinen.

3.2.7.
Gewirre und Gerölle
Texturaddition

An den Texturen, die durch Überlappung von Einzelfiguren zustandekommen, konnte das Wesentliche am Texturbegriff aufgezeigt werden. Es gibt jedoch selbst innerhalb der generativen Graphik noch viele andere Verfahren der Texturgenese. Wie wir schon im Zusammenhang mit der Gradation der Texturen feststellten, eignen sich auch Irrwege zur Erzeugung von Texturen. Bild 37 stellt einen zusammenhängenden Irrweg dar, dessen Elementarstücke Kreisbögen sind. Als Protozeichen treten Sterne mit gebogenen Strahlen und Kreisbogenpolygone auf. Wir bezeichnen Texturen, die eigentlich Linien mit vielen Überkreuzungen sind, jedoch auch solche, die nur so aussehen, als Gewirre. Die Subsumption von Texturen unter einen Klassenbegriff — hier der Fadenknäuel unter den der Gewirre — schreibt den betreffenden Texturen eine gemeinsame makroinnovative Komponente zu. Benennung induziert Makroinnovation.

Es folgt nun der Generator zu Bild 37:

```
(1)   1 'BEGIN' 'COMMENT' LOCKEN.,
      2 'REAL' XM, YM, PI2, DA, DD, R, A, ZW.,
```

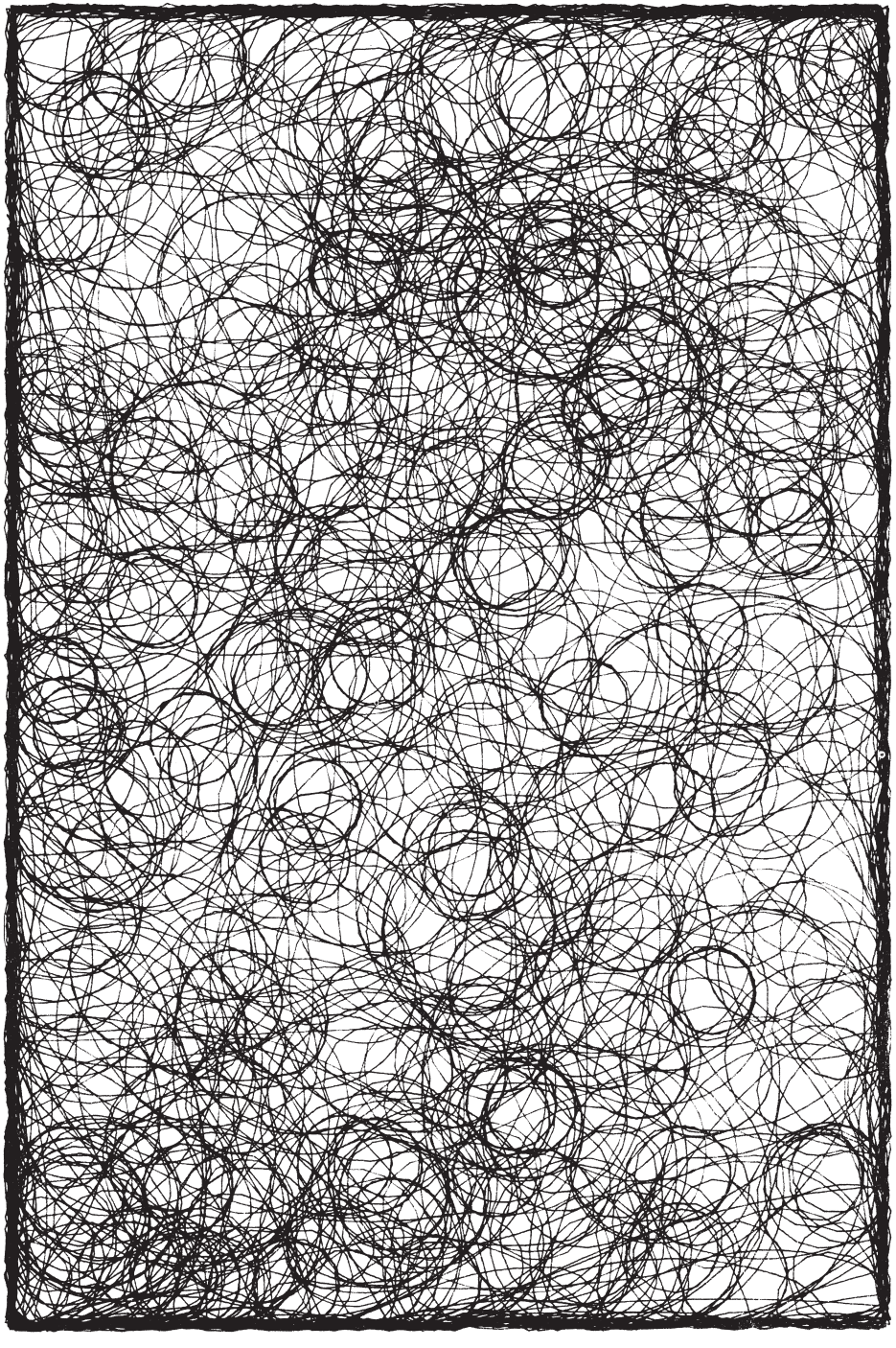


Bild 37

```
 3  'INTEGER' I, K, S, SU, SO.,
 4  OPEN(X,Y)., LEER(0,0).,
 5  PI2.=2*3.14159., DA.=PI2/12.,
    R.=4., A.=0.,
 6  SU.=5., SO.=25., K.=100.,
 7  JI1.=JS1., JI2.=JS4.,
 8  JA1.=-DA., JE1.=DA.,
    JA2.=SU., JE2.=SO.,
 9  'FOR' I.=1 'STEP' 1 'UNTIL' K 'DO'
10  'BEGIN' ZW.=J1., S.=ENTIER(J2).,
11  'FOR'  I.=1 'STEP' 1 'UNTIL' S 'DO'
12          'BEGIN' A.=A+ZW.,
13           DREH..
14           XM.=X+R*COS(A).,
             YM.=Y+R*SIN(A).,
15          'IF' (XM 'GREATER'
             120.0 'OR' XM 'LESS' -120.0
16          'OR' YM 'GREATER'
             80.0 'OR' YM 'LESS' -80.0)
17          'THEN' 'BEGIN' A.=A+DA.,
            'GOTO' DREH 'END'.,
18           LINE(XM,YM)
19          'END'.,
20  'END'., CLOSE
21  'END' LOCKEN.,
```

Der Aufbau dieses Generators ist nicht so kompliziert, als es zunächst den Anschein hat. Bis Doppelzeile 8 reichen die üblichen Vorbereitungen: Festlegung von Konstanten, Anfangswerten, Grenzen von Zufallsgeneratoren. Von Zeile 9 bis 20 erstreckt sich eine Ineinanderschachtelung von zwei Programmschleifen, von denen die äußere die Anzahl der übereinandergezeichneten Kreisbögen, die innere die Anzahl von Elementarstrecken bestimmt, die den einzelnen Kreisbogen auf-

bauen. Die Radien der Kreisbögen hängen vom Zuwachswinkel ZW zwischen je zwei Elementarstrecken ab. In Zeile 10 des Generators wird ZW als Wert des Zufallsgenerators J1 bestimmt; aus den Zeilen 8 und 5 erkennt man, daß ZW höchstens den Betrag von dreißig Winkelgraden haben kann. Die Anzahl S der Elementarstrecken pro Kreisbogen ergibt sich in Programmzeile 10 aus dem Wert des Zufallsgenerators J2 dadurch, daß man eine horizontglobale Standardprozedur ENTIER auf J2 wirken läßt, die zu der zwischen den Grenzen 5 und 25 liegenden reellen Zahl J2 die jeweils nächstkleinere oder gleichgroße ganze Zahl liefert. Ist also etwa J2 gleich 5.67, so ist ENTIER(J2) gleich 5. Eine Besonderheit stellen die Zeilen 15 bis Doppelzeile 17 dar. Sie lenken den Irrweg vom Rand eines Rahmenrechtecks ab, so daß dieser Rand niemals überschritten wird.

Der Absicht des Programmierers gemäß sollte Bild 37 einhundert Kreisbögen enthalten (wobei der einzelne Kreisbogen durch eventuelle Rückbeugungen von der Wand in mehrere Segmente zerbrochen werden kann). Diese Absicht wurde durch einen katastrophalen Programmierfehler durchkreuzt. Betrachtet man nämlich die Zeilen 9 und 11 genauer, so bemerkt man, daß die innere und die äußere Programmschleife durch die gleiche Laufvariable, nämlich I, gezählt werden. Dies hat zur Folge, daß I niemals den für die Beendigung der äußeren Schleife benötigten Wert 100 erreichen kann, weil I in Zeile 11 immer wieder unter den Wert von S herabgedrückt wird, der 25 nicht überschreitet. Die weitere Folge ist, daß das Programm von sich aus überhaupt nicht aufhören kann, zu rechnen, der Zeichenautomat muß von außen abgeschaltet werden. Tatsächlich geschah das im Fall von Bild 37, so diese ästhetische Information einem Zufall zu verdanken ist, der von noch anderer Art ist, als

die Zufallsereignisse, die von den Zufallsgeneratoren ausgelöst werden.

Bild 37 kann natürlich auch von einem semantisch einwandfreien Programm generiert werden, man erhält ein solches zum Beispiel dadurch, daß man in Zeile 3 einen Zähler M hinzufügt, I in Zeile 9 durch M ersetzt und K in Zeile 6 den Wert −1 zuweist. Solche Generatoren, die willkürlich von außen angehalten werden müssen, sind zur Genese von Alphabeten geeignet, die von sich aus eine Gradation aufbauen. Man kann ja beispielsweise den Generator zu Bild 37 noch eine Stunde länger laufen lassen, als im Fall von Bild 37, dann zwei Stunden länger, usw. Dadurch wird man eine Gradation von Graphiken erhalten, deren Mikroinnovation ständig ansteigt, bis das Bild im Chaotischen verschwindet.

Die Mannigfaltigkeit der Texturen ist unübersehbar und eine erschöpfende Strukturtheorie dieser Art von ästhetischer Information vermutlich unmöglich. Wir wollen nur noch zwei weitere Typen von Texturen untersuchen. Der erste Typ wurde schon in Abschnitt 3.2.5 erwähnt, als wir darüber sprachen, daß die meisten vorgefundenen Texturen nicht durch Auf-, sondern durch Abbau- und Zerfallprozesse zustandekommen. Ein Beispiel eines Zerfallsprozesses, der zu typischen Texturen führen kann, ist das sich nach und nach verstärkende Abbröckeln eines regelmäßigen Flächenmusters, wobei wir voraussetzen, daß bei diesem Prozeß die Elementarstücke des Musters nicht weiter zerfallen, sondern erhalten bleiben. Eine Abart des beschriebenen Vorgangs findet sich in der Gesteinskunde, wenn ein kubisch kristallisierendes Mineral in kleinere kubische Stücke zerfällt. Im Rahmen der generativen Graphik liefert Bild 38 ein Beispiel für die Ausbildung einer Textur des Typs, den

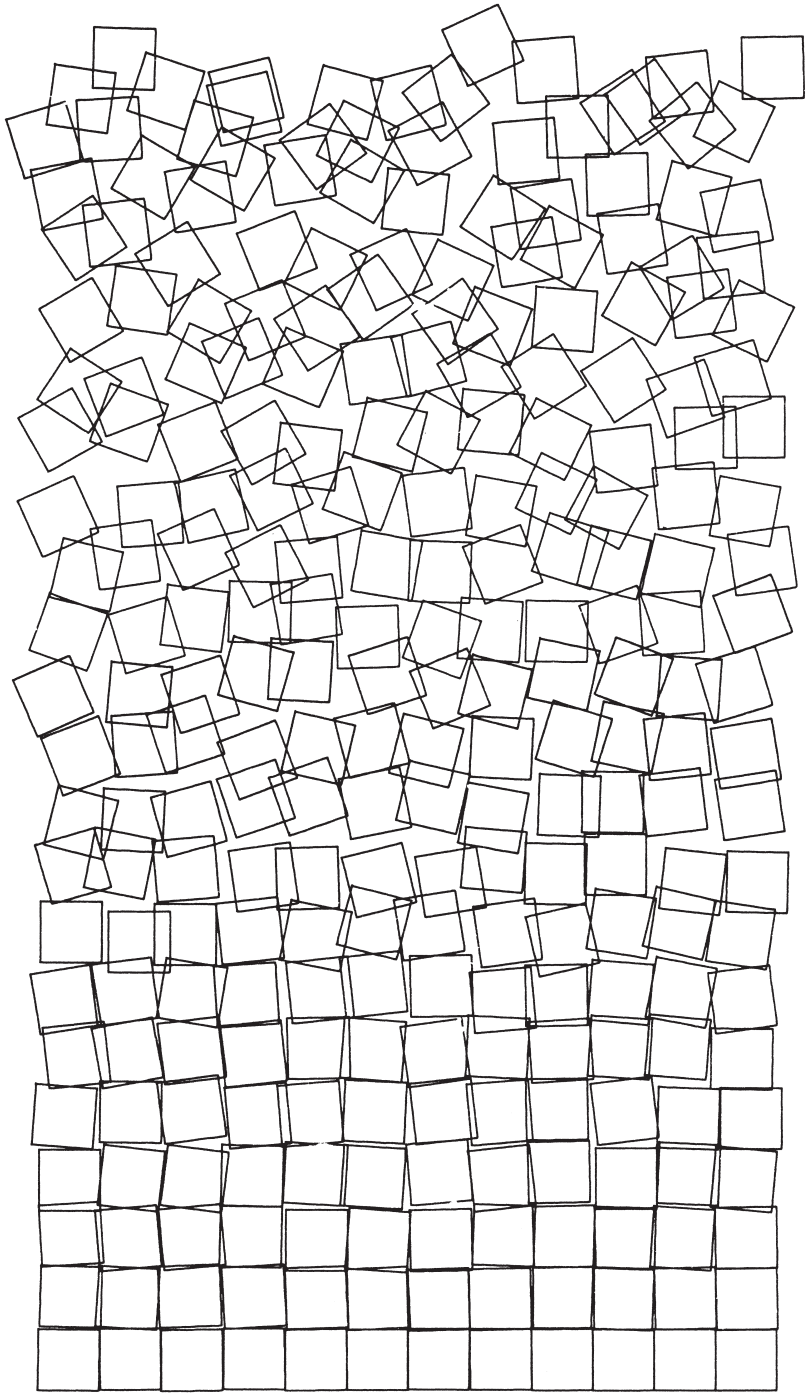
wir gerade besprechen, und den wir als Gerölle oder Schotter bezeichnen wollen. Der Generator für Bild 38 zeigt, wie schrittweise sich verstärkende Zerfallsprozesse mit Hilfe von Programmen simuliert werden können:

(2)

```
 1 'BEGIN' 'COMMENT' SCHOTTER.,
 2 'REAL' R, PIHALB, PI4T.,
 3 'INTEGER' I.,
 4 'PROCEDURE' QUAD.,
 5 'BEGIN'
 6 'REAL' P1, Q1, PSI., 'INTEGER' S.,
 7  JE1.=5*I/264., JA1.=-JE1.,
 8  JE2.=PI4T*(1+I/264).,
    JA2.=PI4T*(1-I/264).,
 9  P1.=P+5+J1., Q1.=Q+5+J1.,
    PSI.=J2.,
10  LEER(P1+R*COS(PSI),
    Q1+R*SIN(PSI)).,
11 'FOR' S.=1 'STEP' 1 'UNTIL' 4 'DO'
12 'BEGIN' PSI.=PSI+PIHALB.,
13  LINE(P1+R*COS(PSI),Q1+R*SIN(PSI))
14 'END'., I.=I+1
15 'END' QUAD.,
16  R.=5*1.4142.,
17  PIHALB.=3.14159*.5., PI4T.=PIHALB*.5.,
18  I.=0.,
19  SERIE(10.0,10.0,22,12,QUAD)
20 'END' SCHOTTER.,
```

Bild 38 wird durch einen Aufruf der Prozedur SERIE erzeugt, die wir schon oft zur Darstellung von Prozeduren herangezogen haben. Zur Genese der Elementarfigur, die in dem von SERIE gesteuerten Kompositionsprozeß vervielfacht wird, dient die parameterlose Prozedur QUAD. In den Zeilen 4 bis 15 des Generators wird QUAD

Bild 38



vereinbart, diese Prozedur zeichnet Quadrate konstanter Seitenlänge, jedoch zufälliger Lage und Winkelstellung. Man erkennt aus den Doppelzeilen 9 und 10, daß die Position des Einzelquadrats vom Zufallsgenerator J1, die Winkellage von J2 beeinflußt wird. Die sukzessive verbreiterte Streuung der relativen Ortskoordinaten P und Q und des Lagewinkels PSI des einzelnen Quadrats wird durch einen Zähler I gesteuert, der bei jedem Aufruf von QUAD weitergeschaltet wird (siehe Zeile 14).

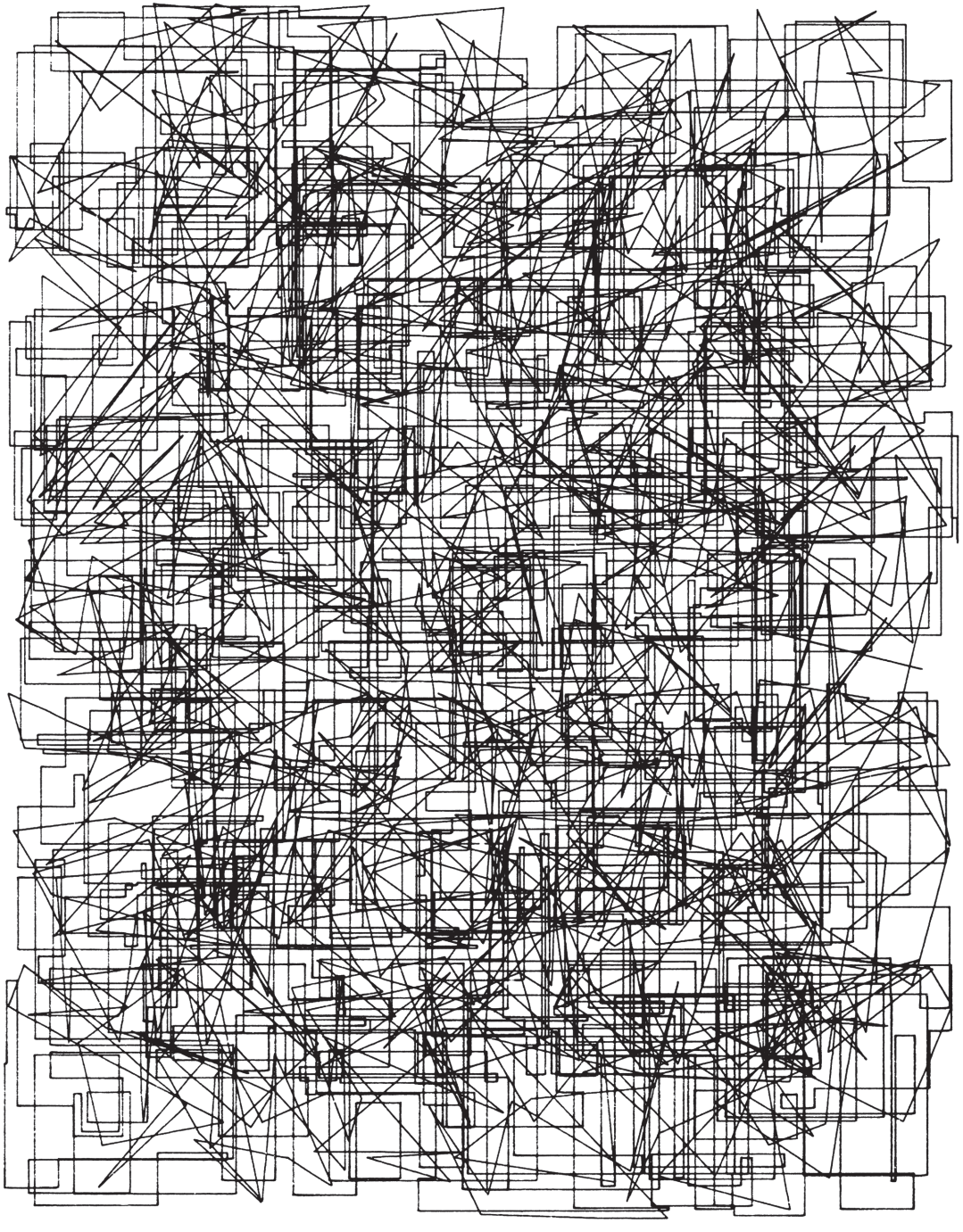
Schottertexturen und Haufenbilder haben miteinander zu tun, und zwar können die Schotter als Grenze einer Gradation gedeutet werden, die mit örtlich weit gestreuten Haufen einsetzt. Schrittweise Vermehrung der Individuen in diesen Haufen, bei konstant gehaltenen Streugrenzen, führt schließlich zu Überlappungseffekten und damit zur Texturentstehung.

Wir kommen nun zu dem zweiten Texturtyp. Man kann neue Texturen natürlich auch dadurch gewinnen, daß man alte Texturen überlagert oder addiert. Freilich kann ein Gewinn an endogener Innovation bei der Addition von Texturen nur durch zusätzliche Überlappungseffekte zustandekommen, die bei der Addition auftreten. Bild 39 zeigt die Addition der Bilder 29 und 30. Der Lokalnexus von Bild 39 ist jedoch in dem Sinn labil, daß er bei der Perzeption zu einem Zerfall in die Lokalnexus der addierten Bilder neigt. Es ist jedoch sicher, daß Texturen aufgefunden werden können, die — einmal addiert — keine perzeptorische Trennung mehr gestatten.

3.2.8. Anisotropie in Texturen

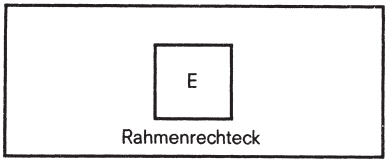
Wir begegneten dem Anisotropiebegriff schon in Abschnitt 3.2.4, wo durch Anisotropisierung, d. h. durch

Bild 39



Übertreibung einer Ausdehnungsrichtung aller Einzelfiguren in einem Variationenbild, die Makroinnovation einer Bildinformation steigern konnten. Die Bilder 24 und 25 stellen anisotrope Variationen dar. In Abschnitt 3.2.5 haben wir gelernt, Texturen durch die gegenseitige Verhakung der Einzelfiguren von Variationen zu erzeugen. Der Überlappungseffekt, der bei diesem Verfahren auftritt, ist verantwortlich für das Zustandekommen des typischen Lokalnexus der entstehenden Textur. Nun ist es naheliegend, sich zu fragen, welcher Texturtyp denn durch Verhakung von anisotropen Variationen zustandekommen würde, ja, ob man so erzeugten Bildinformationen Texturcharakter überhaupt noch mit Recht zuschreiben dürfte?

Das Verhakungsverfahren zur Texturgenese haben wir mit Hilfe der Prozedur SERIE zustandegebracht. Wir erinnern uns, daß die Überlappung der Einzelfiguren einer Variation, die durch einen Aufruf von SERIE hervorgerufen wird, von der gegenseitigen Lage des Elementarrechtecks von SERIE und des Rahmenrechtecks der Einzelfigur abhängt. Das Elementarrechteck von SERIE ist dabei das Elementarfeld der durch SERIE vorgenommenen schachbrettartigen Zerlegung der Zeichenfläche. Eine typische Gegeneinanderlagerung von Elementarrechteck und Rahmenrechteck zeigt Figur 6 in Abschnitt 3.2.4. Es ist zu vermuten, daß man Anisotropisierungswirkungen bei der Anwendung von SERIE leicht dadurch erzielen kann, daß man den Seitenverhältnissen der an Figur 6 beteiligten Rechtecke besondere Werte zuweist. Figur 8 zeigt ein Beispiel, das zur Genese der Bilder 40 und 41 herangezogen wurde.



Figur 8

E = Elementarrechteck

Zur Erzeugung der Bilder 40 und 41 können wir das Schema des Programms (1) aus Abschnitt 3.2.5 verwenden, das zur Erzeugung der Bilder 29 und 30 benutzt wurde. Wir brauchen die Zeilen dieses Programms lediglich so abzuändern, daß an die Stelle von Figur 6 aus Abschnitt 3.2.4 jetzt Figur 8 tritt. Das neue Programm lautet folgendermaßen:

```
(1)   'BEGIN' 'COMMENT' ANISOTROPIE
      0 UND 1.,
      JI1.=JS2., JI2.=JS5.,
      N.=10.,
      JLI.=-20., JRE.=30.,
      JUN.=-5., JOB.=15.,
      T.=0.,
      SERIE(10.0,10.0,16,12,ELIRR)..,
      T.=1.,
      SERIE(10.0,10.0,16,12,ELIRR)..,
      'END' ANISOTROPIE 0 UND 1.,
```

Die Schlankheit des Rahmenrechtecks gegenüber dem Elementarrechteck in Figur 8 gibt bereits eine Vorstellung von der Vorzugsrichtung, die sich in der Struktur der generierten Bilder einstellen muß, die ja nichts anderes als die Überlagerungen der durch die Prozedur

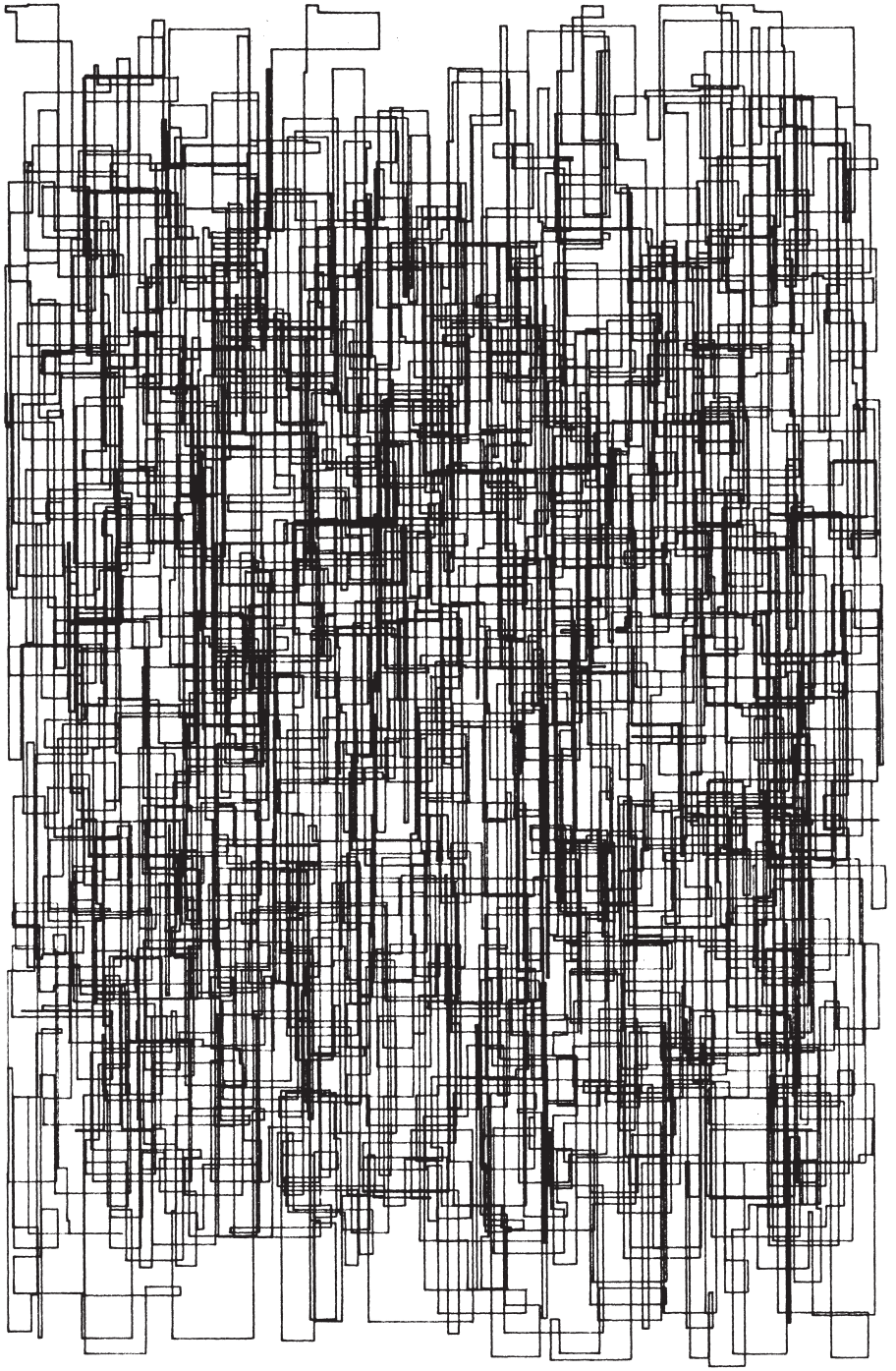
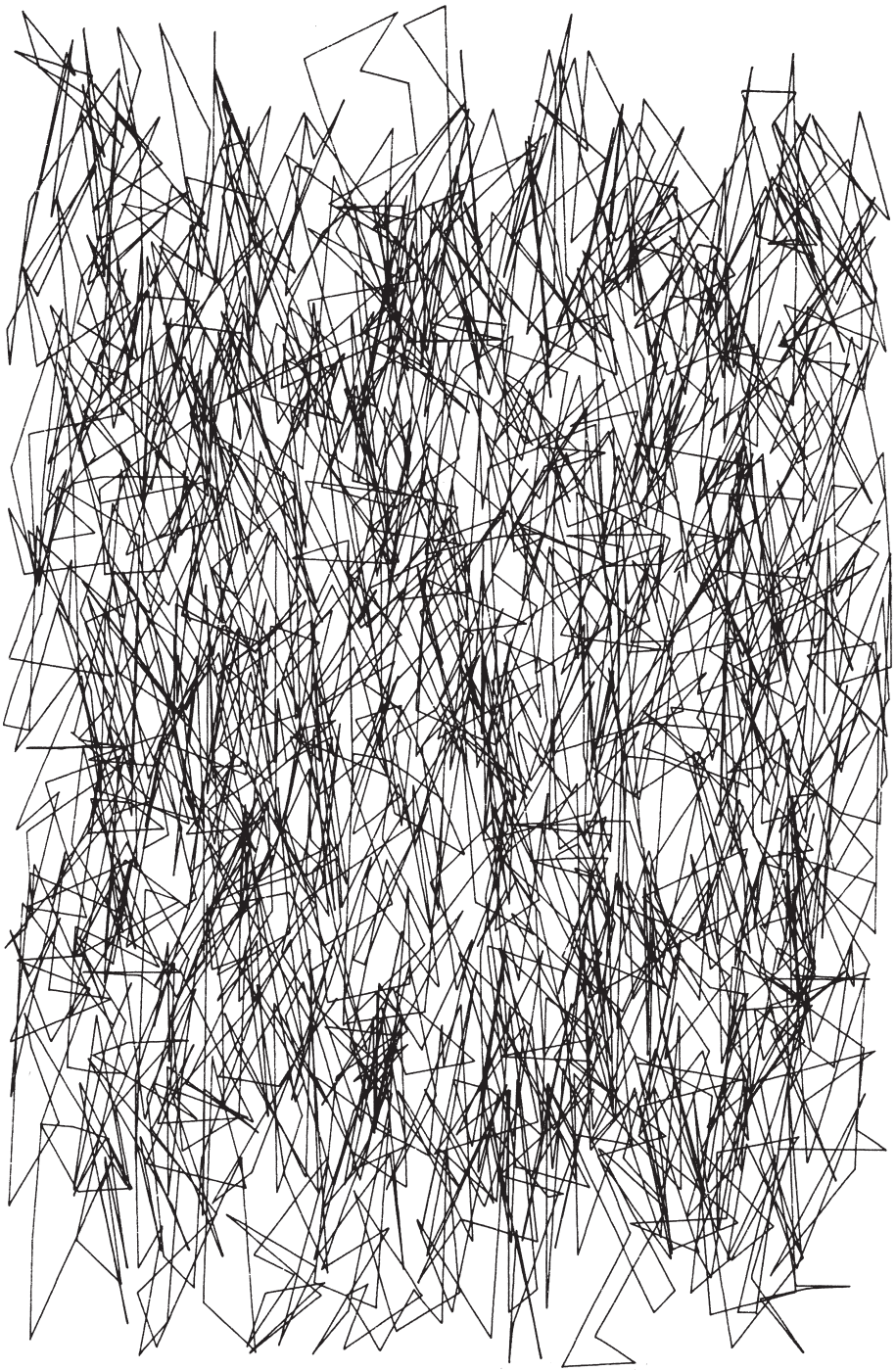


Bild 40

Bild 41



ELIRR in das Rahmenrechteck eingezeichneten elementaren Irrwege sind.

Ein zweiter Weg zur Gewinnung von Anisotropie in Texturen führt zwar ebenfalls über die Prozedur SERIE, benutzt jedoch nicht ELIRR. Wir gehen von dem Generator MIKADOSPIELHAUFEN I aus, den wir in Abschnitt 3.2.5 unter (3) finden. Wenn wir dort Zeile 9 derart abändern, daß wir JA1 gleich PI/6 und JE1 gleich JA1+PI/30 setzen, so erhalten wir einen neuen Generator, in dem die Streugrenzen für die Winkellagen der durch die Prozedur STAB erzeugten Strecken wesentlich vermindert sind. Das Streuintervall betrug 180 Winkelgrad im Fall von Bild 34, umfaßt jedoch nur noch 12 Winkelgrad im Fall von Bild 42.

Der typische Nexus von Bild 42 kommt, ebenso wie die Nexus der beiden vorhergehenden Bilder, durch den Überlappungseffekt zustande. Wir kehren nun zu der Frage zurück, die wir schon am Anfang dieses Abschnitts gestellt haben, ob nämlich anisotropisierte Texturen überhaupt noch als Texturen im strengen Sinn zu betrachten seien. Ein Ansatz, der zu einer Beantwortung dieser Frage führen könnte, wurde schon am Schluß des Absatzes formuliert, der in Abschnitt 3.2.5 auf die Abfassung des Generators (3) folgt: *Man erkennt, daß man die Makroinnovation in Bild 34 durch Einengung der Streugrenzen von J1 steigern könnte.* Diese Bemerkung ist folgendermaßen einsichtig zu machen: Zunächst bedingt die Einengung der Streugrenzen von J1 eine Verminderung der Mikroinnovation der Bildinformation, was man daran sieht, daß die mikroinnovative Wirkung des Zufallsgenerators J1 ganz eliminiert wird, wenn man das Streuintervall von J1 von 12 Winkelgraden vollends auf den Wert null herabdrückt. Ein solcher Eingriff von

Bild 42

außen setzt induzierte Effekte, wobei zunächst noch nicht bewiesen ist, daß die hervorgerufenen Effekte eine Erhöhung der Makroinnovation des Bildes bedingen. Texturen, deren wesentliche Information aus dem mikroinnovativ aufgebauten Lokalnexus und der Anordnung der Protozeichen in ihm stammt (siehe Abschnitt 3.2.6), sind empfindlich störbar durch wesentliche Eingriffe in die Funktion der Zufallsgeneratoren. Derartige Störungen könnten aber immer noch lediglich eine Abänderung der Art des Lokalnexus bewirken, wenn sich mehrere gleichzeitig abgeänderte Zufallsgeneratoren in ihrer Wirkung gegenseitig ergänzen. Wird jedoch die Wirkungsmöglichkeit eines Zufallsgenerators eingeschränkt, ohne daß geeignete andere dieser Einschränkung gleichfalls unterworfen werden, oder wird die Wirkungsmöglichkeit weiterer Zufallsgeneratoren in bestimmter Weise verändert, so kann sich ein Wechsel in der Struktur der Gesamtinformation des Bildes einstellen, der über Artänderungen des Lokalnexus oder Zu- bzw. Abnahmen der lokalen Innovationsdichte hinausgeht. Dann tritt eine Zusatzinformation auf, die sich nicht nur lokal bemerkbar macht, sondern voneinander entfernte Orte im Bild miteinander verknüpft. Diese Zusatzinformation entsteht durch Eingriff von außen, sie enthält also auf jeden Fall exogene Makroinnovation. Da sie jedoch von Überlappungseffekten mitbestimmt wird, enthält sie auch endogene Anteile. Sicher ist, daß die Zusatzinformation ein Mehr an Makroinnovation bedeutet, d. h. einen Zuwachs an Makroinnovation. Genau dieser Fall tritt beim Übergang von Bild 34 zu Bild 42 ein.

Anisotropisierung durch bestimmte Änderungen der Streugrenzen von Zufallsgeneratoren — jedes der drei Bilder 40, 41 und 42 ist davon betroffen — hat also eine Erhöhung der Makroinnovation der Bildinformation zur

Folge. Nun wieder zurück zu unserem Hauptproblem: *Sind anisotropisierte Texturen überhaupt noch Texturen?* In Abschnitt 3.2.5 kennzeichneten wir die Texturen dadurch, daß wir ihnen *Lokalnexus ohne Distalnexus* zuschrieben. Durch das Auftreten von zusätzlicher Makroinnovation, die sich aus Bildelementen aufbaut, die nicht unmittelbar benachbart sind, stellt sich Nexus über Distanz, d. h. Distalnexus, her. Daraus folgt, daß durch Anisotropisierung die angeführte Kennzeichnung der Texturen ungültig wird.

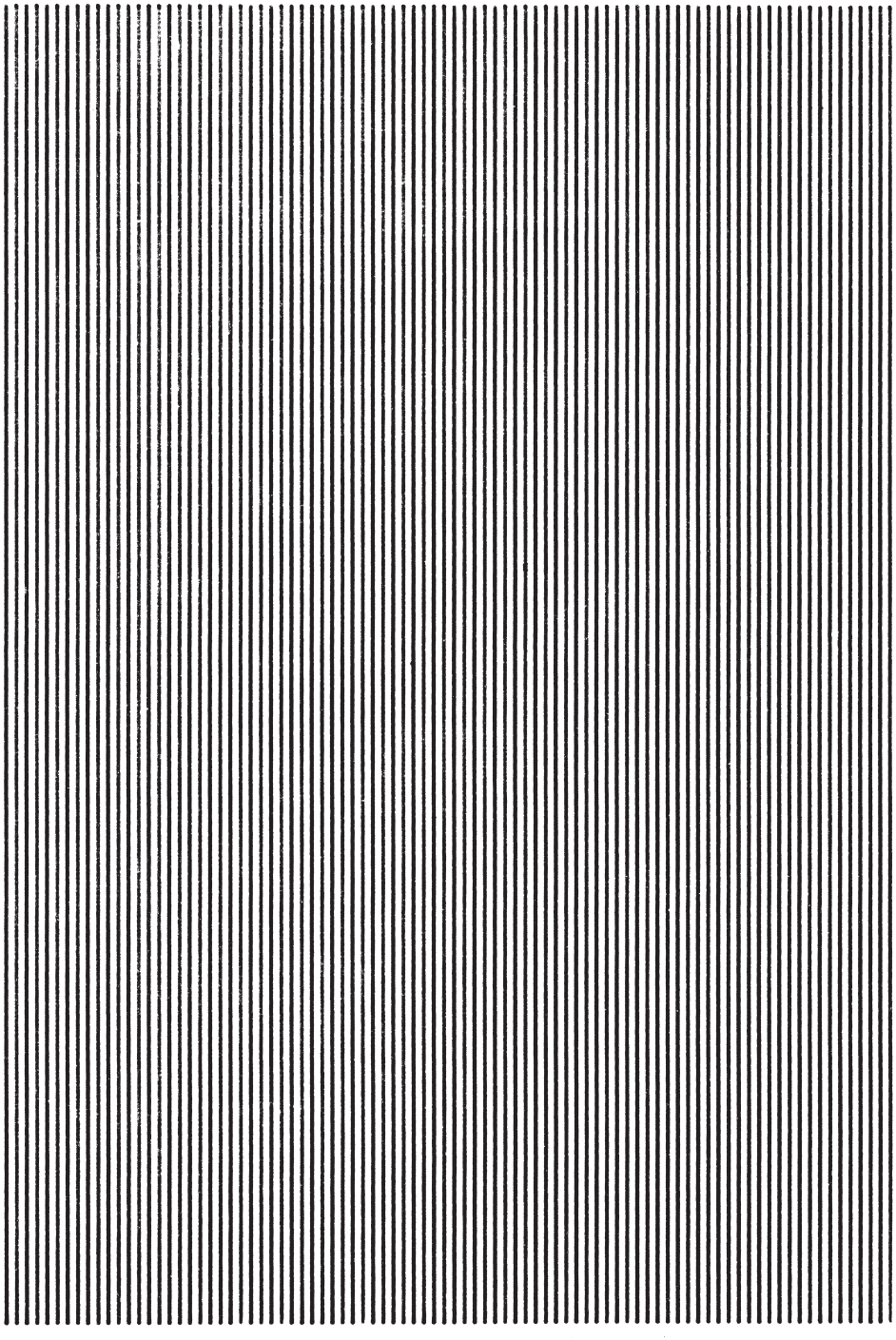
Anschaulich wirkt sich die Anisotropisierung von Texturen durch das Zustandekommen einer Vorzugsrichtung in der Textur aus. Die Textur ist plötzlich keine Wüstenei mehr, in der man sich verlaufen könnte. Man braucht nur der Vorzugsrichtung zu folgen, um an einen möglichen Rand des Texturfeldes zu gelangen. Die Anisotropie überbrückt eben Entfernungen. Andererseits bleibt eine wichtige Eigenschaft der Texturen auch bei Anisotropisierung erhalten: Die lokale Beschaffenheit ist überall die gleiche, wenn auch nicht identisch von einer Stelle an die andere verschieblich. Wir tragen der Tatsache, daß die durch Anisotropisierung aus Texturen entstehenden Bildinformationen wenigstens teilweise Texturcharakter haben, dadurch Rechnung, daß wir sie per definitionem zu den Texturen zählen. Anders ausgedrückt, hat uns der Aufbau von Distalnexus durch die Anisotropisierung dazu geführt, unsere Definition der Textur zu erweitern.

Anisotropisierung vermehrt also Makroinnovation, vermindert Mikroinnovation. Dadurch erhalten wir eine Gradation, die bei Texturen ohne Distalnexus einsetzt und bei stark anisotropen Texturen endet. Genau gesagt muß das Ende der Gradation durch das Verschwinden

der Mikroinnovation gekennzeichnet sein — dementsprechend der Anfang durch das Verschwinden der Makroinnovation. Setzen wir den Anfang dieser Gradation gleich mit dem nebulosen Chaos, wie sieht dann das Ende aus?

Der Kybernetiker McKay hat Bildinformation untersucht, die sich durch extreme Regelmäßigkeit über das gesamte Bildfeld hinweg auszeichnet. Für den Menschen ist die Perzeption McKay'scher Bilder ausgesprochen unangenehm. Wir bringen ein Beispiel eines solchen Bildes, das mit dem Zeichenautomaten hergestellt wurde, dessen Vorbild sich jedoch in dem Buch *Auge und Gehirn* von R. L. Gregory (München, 1966) befindet. Gregory berichtet über McKay's Versuche. Bei längerer Betrachtung von Bild 43 sieht man eine rein im Perzeptor entspringende Sekundärtextur: Wellige Linien. McKay nimmt an, daß das visuelle System durch die ständige Wiederholung des gleichen Musters gestört wird. Die Redundanz ist für das visuelle System zu hoch, die Wiederholung der Innovation so eintönig (die pure Makroinnovation ist ja ein einfacher senkrechter Strich), daß der Perzeptor von sich aus Mikroinnovation addiert, um seinen inneren Informationsumsatz in den für ihn annehmbaren Grenzen zu halten. Max Bense nähert sich diesem Tatbestand von ästhetischen Überlegungen aus und in AE, Seite 217, heißt es: „Nur vom Begriff der Redundanz her, die die restlos unwahrscheinliche Verteilung der Zeichen im Zeichenfluß, also die maximale ästhetische Information verhindert, kann das ideale Kunstwerk äußerster Perfektion vom realen Kunstwerk unvermeidlicher Dispersion unterschieden werden." In Bild 43 ist die Mikroinnovation gleich null, Wirkungen von Zufallsgeneratoren scheiden aus, das Bild ist deshalb auch kein Fall unter vielen gleichartigen, unter de-

Bild 43



nen es mit einer gewissen Wahrscheinlichkeit aufträte — die Bildinformation ist restlos unwahrscheinlich. Im Sinne der Bemerkung von BENSE ist Bild 43 das Modell eines perfekten Kunstwerks.

Die Texturen sind also ausgespannt zwischen Grenzfällen ästhetischer Information, die beide für die Unterhaltung eines vitalen Informationsflusses — für den Ablauf von Kommunikation — unbrauchbar sind: Sie sind ausgespannt zwischen Chaos und Nichtssagendem.

Wenn wir am Schluß von Abschnitt 3.2.6 sagten, daß die Textur im mittleren Bereich einen Grad der Wiederholung von Innovativem aufzeigt, dem eine gewisse Angepaßtheit des Perzipienten entspricht, und wenn wir eben feststellten, daß die Wahrnehmbarkeit einer Textur mit der Unterhaltung eines vitalen Informationsflusses zusammenhängt, so klingen hier Gedanken an, die A. A. MOLES und nach ihm H. FRANK erörtert haben. MOLES sagt nämlich, „daß der Informationsfluß eines zeitlichen Kunstwerks eine obere Grenze haben müsse, um noch wahrnehmbar zu sein, aber auch eine untere Grenze, damit es nicht zu informationsarm, also banal und langweilig ist" (MOLES 1958, zitiert in H. FRANK: Kybernetische Analysen subjektiver Sachverhalte, Quickborn bei Hamburg 1964, Seite 36; FRANK fährt a. a. O. mit interessanten quantitativen Überlegungen fort). Auch unsere Zusatzbemerkung zur MCKAYSCHEN Figur, Bild 43, daß nämlich der Perzeptionsapparat die Redundanz durch Addition einer Sekundärtextur ergänze, um seinen inneren Informationsumsatz in den für ihn annehmbaren Grenzen zu halten, gehört in diesen Zusammenhang.

3.2.9. Gewebe

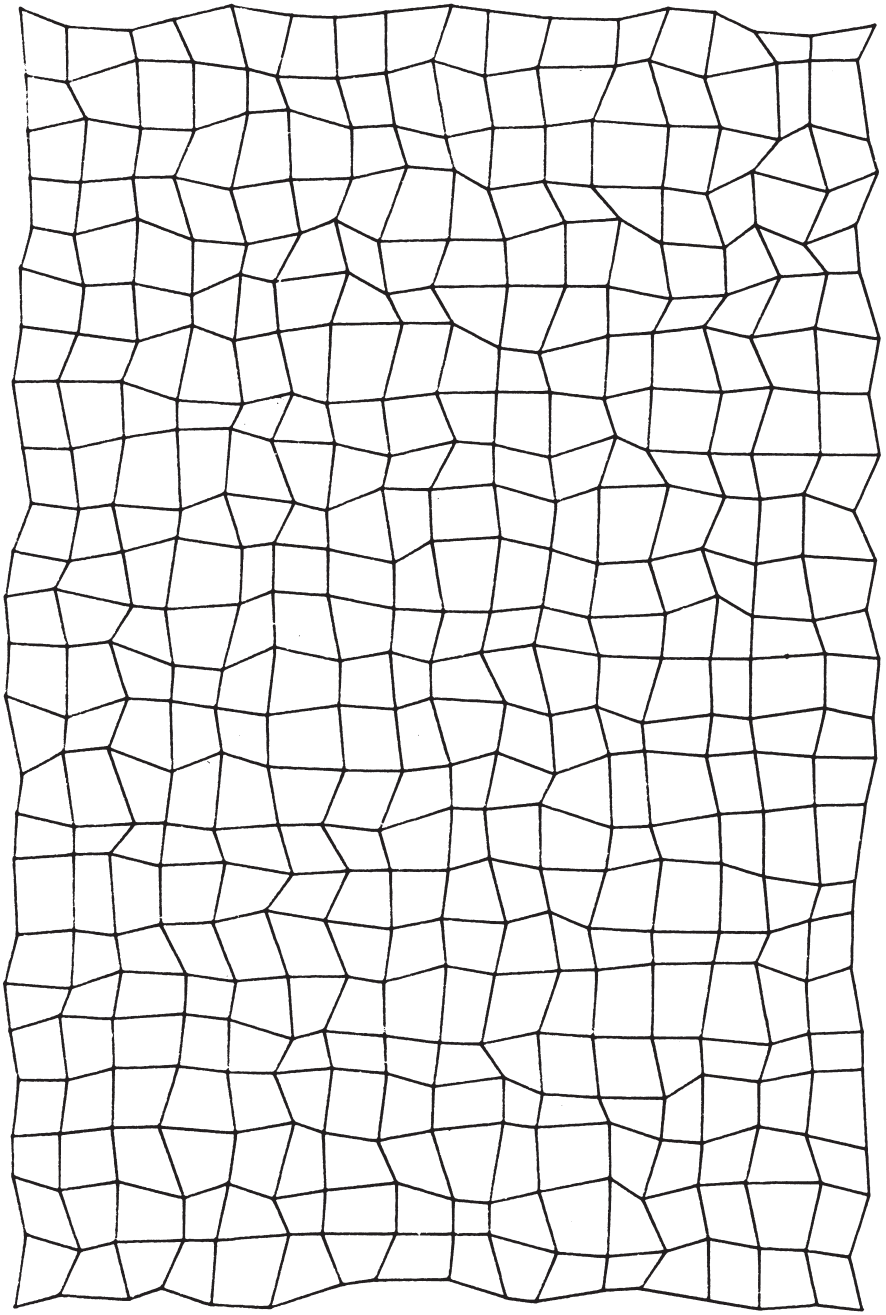
Am Anfang von Abschnitt 3.1.4 bezeichneten wir den Generator einer Graphik als ein Modell des künstlerischen Schöpfungsprozesses. Freilich sind alle so verstandenen Modelle, die in der vorliegenden Arbeit untersucht werden, äußerst einfach aufgebaut, sie sind extreme Simplifikationen. Unter den vielen Eigentümlichkeiten des künstlerischen Schöpfungsprozesses, die wir bis jetzt nicht modelliert haben, findet sich auch das Merkmal, daß spätere Stadien der Entstehung eines Kunstwerkes von früheren beeinflußt werden. Z. B. sind später generierte Details im allgemeinen von früher gesetzten Schwerpunkten in einem Bild abhängig. Will man den Einfluß früherer Genesestadien auf spätere im Maschinenmodell erfassen, so wird die Konstruktion eines Modells im Modell insofern notwendig, als der Generator ja ein Modell der entstehenden Zeichenkonstellation speichern muß, um sich den ständigen Zugriff auf früher entstandene Bildelemente zu sichern. Diese Vorstellung trifft ohne Zweifel auf den Maler vor der Staffelei zu: Introspektiv einsehbar ist, daß er frühen Bildeinzelheiten Individualität zuordnet, die er im Gedächtnis behält. Offenbar wird eine derartige Informationszwischenspeicherung beim Trancemalen (und vermutlich auch bei der Jazzimprovisation) nur vermindert vorgenommen.

Bei der Programmierung *speicherabhängiger Information* muß also das Programm, das den Generator darstellt, Merkmale von Bildeinzelheiten so speichern, daß sie später wieder aufrufbar, d. h. aktualisierbar sind. Auf der Basis dieses Prinzips eine allgemeine Programmiertheorie zu entwickeln, wäre nützlich und eigentlich notwendig, würde aber den Rahmen der vorliegenden Untersuchungen sprengen. Wir wollen jedoch an einem Beispiel zeigen, wie Merkmalszwischenspeicherung funktionieren kann.

Sind die zu speichernden Bildeinzelheiten regelmäßig angeordnet, so liegt es nahe, sie in Tabellenform abzuspeichern. In der Mathematik ist anstatt *Tabelle* der Ausdruck *Matrix* gebräuchlich und die einzelnen Größen in einer Matrix werden durch Anhangszahlen, auch Indizes genannt, unterschieden. Betrachten wir nun das Beispiel näher, das wir programmieren und generieren wollen: Es wird in einem Prototyp durch Bild 44 dargestellt! Denkt man sich, das Bild im Breitformat betrachtet, zunächst alle (annähernd) horizontalen Linien fort, so bleiben vertikale, offene Polygonzüge übrig. Eine Bedingung der Bildgenese besteht nun darin, die zunächst als nichtexistent vorgestellten horizontalen Polygonzüge so durch die Eckpunkte der vertikalen zu legen, daß eine Rasteranordnung entsteht. Es leuchtet sofort ein, daß die Genese der Steuerinformation für die horizontalen Züge auf die Eckpunktkoordinaten der vertikalen Züge zurückzugreifen hat. Nun wenden wir die vorhin entwickelte Matrizenvorstellung an. Wenn es gelänge, von Anfang an alle Eckpunktkoordinaten überhaupt in der Form von Matrizenelementen $A_{p,q}$ und $B_{p,q}$ zu speichern, wobei die $A_{p,q}$ die Abszissen, die $B_{p,q}$ die Ordinaten der Eckpunkte darstellen, so wäre das Geneseproblem gelöst, da dann alle gewünschten Eckpunktkoordinaten beliebig zur Verfügung stünden (so daß man z. B. zusätzlich noch Diagonalen in das Raster legen könnte).

Matrizen und Indizes haben wir zwar bis jetzt noch nicht im Zusammenhang mit der Konstruktion von Generatoren betrachtet, sie sind aber erfreulicherweise Elemente der Sprache ALGOL. Da die Tiefstellung der Indizes den Eingabetastaturen der Datenverarbeitungsanlagen Schwierigkeiten macht, wird die Matrizenschreibweise im ALGOL-System geeignet abgeändert. Man schreibt nämlich z. B. an Stelle von $A_{p,q}$ den Ausdruck A(/P,Q/).

Bild 44



Hat man diese Schreibweise zur Verfügung, so kann man indizierte Größen ebenso wie andere Namen in Wertzuweisungen verwenden, so daß z. B. geschrieben werden darf: A(/P,Q/).=7+P+J2. Durch diese Wertzuweisung wird dem Matrizenelement A(/P,Q/) der Wert zugewiesen, den der arithmetische Ausdruck 7+P+J2 in diesem Augenblick hat.

Wir erinnern uns, daß alle in einem Programm verwendeten Größen am Anfang des Programms vereinbart werden müssen (siehe Abschnitt 2.1.2). Dies gilt auch für indizierte Größen. Als Wortsymbolfolge, die den zu vereinbarenden indizierten Größen vorangeht, dient 'REAL' 'ARRAY'. Auf 'REAL' 'ARRAY' folgen die Namen der indizierten Größen und eine Angabe, welchen Zahlenbereich die Indizes jeweils überstreichen. Besteht die Matrix A(/P,Q/) z. B. aus 25 mal 17 = 425 reellen Zahlen, so ist 'REAL' 'ARRAY' A(/−12..12,−8..8/) eine gültige Vereinbarung. Ausdehnungsgleiche Matrizen dürfen in einer Vereinbarung beliebig vor der numerischen Ausdehnungsangabe aufgereiht werden.

Wir bringen nun eine Prozedur, die Bild 44 und andere derartige Raster zu generieren fähig ist:

(1)

```
 1 'PROCEDURE' GEWEBE(S).,
 2 'VALUE' S., 'REAL' S.,
 3 'BEGIN' 'INTEGER' P, Q.,
 4  JA1.=JA2.=−S., JE1.=JE2.=S.,
 5 'FOR' P.=−12 'STEP' 1 'UNTIL' 12 'DO'
 6 'BEGIN'
 7 'FOR' Q.=−8 'STEP' 1 'UNTIL' 8 'DO'
 8 'BEGIN'
 9  A(/P,Q/).=7*P+J1., B(/P,Q/).=7*Q+J2
10 'END'
```

```
11  'END'.,
12  'FOR' P.= −12 'STEP' 1 'UNTIL' 12 'DO'
13  'BEGIN'
14   LEER(A(/P,−8/),B(/P,−8/)).,
15  'FOR' Q.= −7 'STEP' 1 'UNTIL' 8 'DO'
16   LINE(A(/P,Q/),B(/P,Q/))
17  'END'.,
18  'FOR' Q.= −8 'STEP' 1 'UNTIL' 8 'DO'
19  'BEGIN'
20   LEER(A(/12,Q/),B(/12,Q/)).,
21  'FOR' P.=11 'STEP' −1 'UNTIL' −12 'DO'
22   LINE(A(/P,Q/),B(/P,Q/))
23  'END'
24  'END' GEWEBE.,
```

Die Prozedur (1) nennen wir GEWEBE, weil Bild 44 eine gewisse Ähnlichkeit mit einem Gewebe hat: Man kann die Vertikallinien mit Ketten-, die Horizontallinien mit Schußfäden vergleichen. Der Körper der Prozedur GEWEBE stützt sich auf die vorauslaufende Vereinbarung 'REAL' 'ARRAY' A,B(/−12..12,−8..8/) der zwei Matrizen A und B mit je 425 Elementen. Zu jedem Kreuzungspunkt des Gewebes gehören zwei Indizes P und Q mit Werten zwischen −12 und 12, bzw. −8 und 8, derart, daß A(/P,Q/) die Abszisse und B(/P,Q/) die Ordinate des betreffenden Kreuzungspunktes ist. Die Koordinatenmatrizen A und B sind also in GEWEBE horizontglobal, wie wir uns früher ausdrückten. GEWEBE besitzt einen Parameter S, dessen Bedeutung wir aus dem Aufbau dieser Prozedur erschließen werden.

In Zeile 4 von (1) wird beiden Zufallsgeneratoren J1 und J2 der Zahlenbereich von −S bis S als Streuintervall zugeordnet. Von Zeile 5 bis Zeile 11 erstreckt sich nun die Schachtelung von zwei Laufanweisungen (eine derartige

Schachtelung lernten wir bereits in Abschnitt 2.1.3 kennen), deren Laufkern allen Kreuzungspunkten des Gewebes Koordinatenwerte zuweist. Wir müssen etwas genauer betrachten, wie das vor sich geht: Die Anweisungen des Laufkerns stehen beide in Zeile 9 und sie verwenden die Werte der Indizes P und Q unmittelbar zur Erzeugung der eigentlichen Koordinatenwerte. Dabei wird die Tatsache benutzt, daß die Indexwerte, die ja zwischen −12 und 12 bzw. −8 und 8 liegen, sich der zentralen Lage des Koordinatenursprungs in Zeichenblattmitte anpassen. Lediglich ein von der Größe des Originalzeichenblattes abhängiger Maßstabsfaktor mit dem Wert 7 wird noch mit P und Q multiplikativ verknüpft. Viel wichtiger ist aber, daß zu den so entstehenden Koordinatenwerten 7 mal P bzw. 7 mal Q noch die Zufallszahlen J1 bzw. J2 addiert werden. Das bedeutet jedoch nichts anderes, als daß die Kreuzungspunkte des Gewebes um einen Störsummanden J1 bzw. J2 verändert, d. h. aus der Lage herausgehoben werden, die einem reinen Quadratraster entspricht. Jetzt verstehen wir auch die Bedeutung des Parameters S der Prozedur GEWEBE(S): S ist der Maximalbetrag des Störsummanden, um den jeder Kreuzungspunkt des Gewebes exzentrisch zu einer symmetrischen Ideallage versetzt sein kann.

Durch die zwei lapidaren Aufrufe GEWEBE(2.0) und GEWEBE (10.0) werden die Bilder 44 und 46 generiert. In Bild 46 kann jeder Kreuzungspunkt um maximal 10 Längeneinheiten in jeder Koordinatenrichtung von der Ideallage weggeschoben sein. Die Störung ist also in Bild 46 wesentlich größer als in Bild 44, obgleich auch in Bild 46 an vielen Stellen noch die fest verschweißten Kreuzungspunkte aufzufinden sind. Während Bild 46 eindeutig den Charakter einer Textur hat, liegt Bild 44 in der

Nähe der Ornamente (siehe nächsten Abschnitt 3.2.10). Bild 46 kennzeichnet den Zufall als *Durcheinanderwerfer,* dem in diesem Bild um das Fünffache mehr Raum gegeben wurde als in Bild 44.

Ein reizvolles Problem stellt sich durch die Forderung, zwischen Bild 44 und Bild 46 eine ästhetische Information einzuschieben, die zwar stärker als Bild 44 gestört ist, jedoch nicht einfach mikroästhetisch durch einen anderen Streubereich der Zufallsgeneratoren verändert ist, sondern durch die Setzung eines induzierten makroinnovativen Effekts. Man kann dieses Problem dadurch lösen, daß man den Störeffekt von der jeweiligen Entfernung des Kreuzungspunktes vom Koordinatenursprung abhängig macht. Generator (2) für Bild 45 zeigt die Anwendung eines solchen Verfahrens.

(2)

```
'BEGIN'
'COMMENT'
GEWEBE STOERUNG ZENTRIERT.,
'REAL' NK, ZK, F.,
'INTEGER' P, Q.,
'REAL' 'ARRAY' A, B(/−12..12,−8..8/).,
JA1.=JA2.=−1.,
JE1.=JE2.=1.,
JI1.=JS3., JI2.=JS4.,
NK.=5.,
ZK.=200.,
'FOR' P.=−12 'STEP' 1 'UNTIL' 12 'DO'
'BEGIN'
'FOR' Q.=−8 'STEP' 1 'UNTIL' 8 'DO'
'BEGIN'
F.=ZK/(7*SQRT(P*P+Q*Q)+NK).,
A(/P,Q/).=7*P+J1*F.,
B(/P,Q/).=7*Q+J2*F.,
```

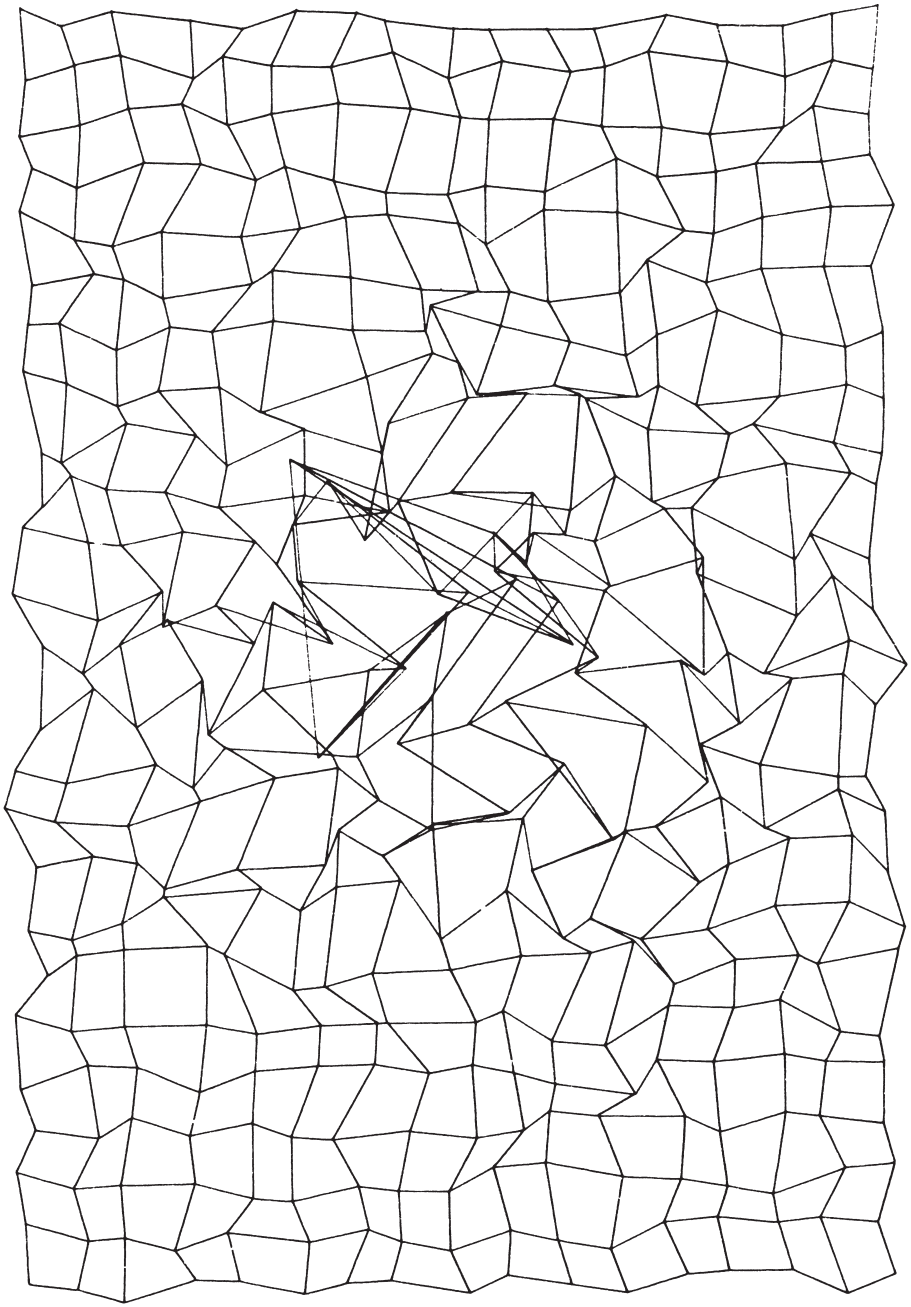
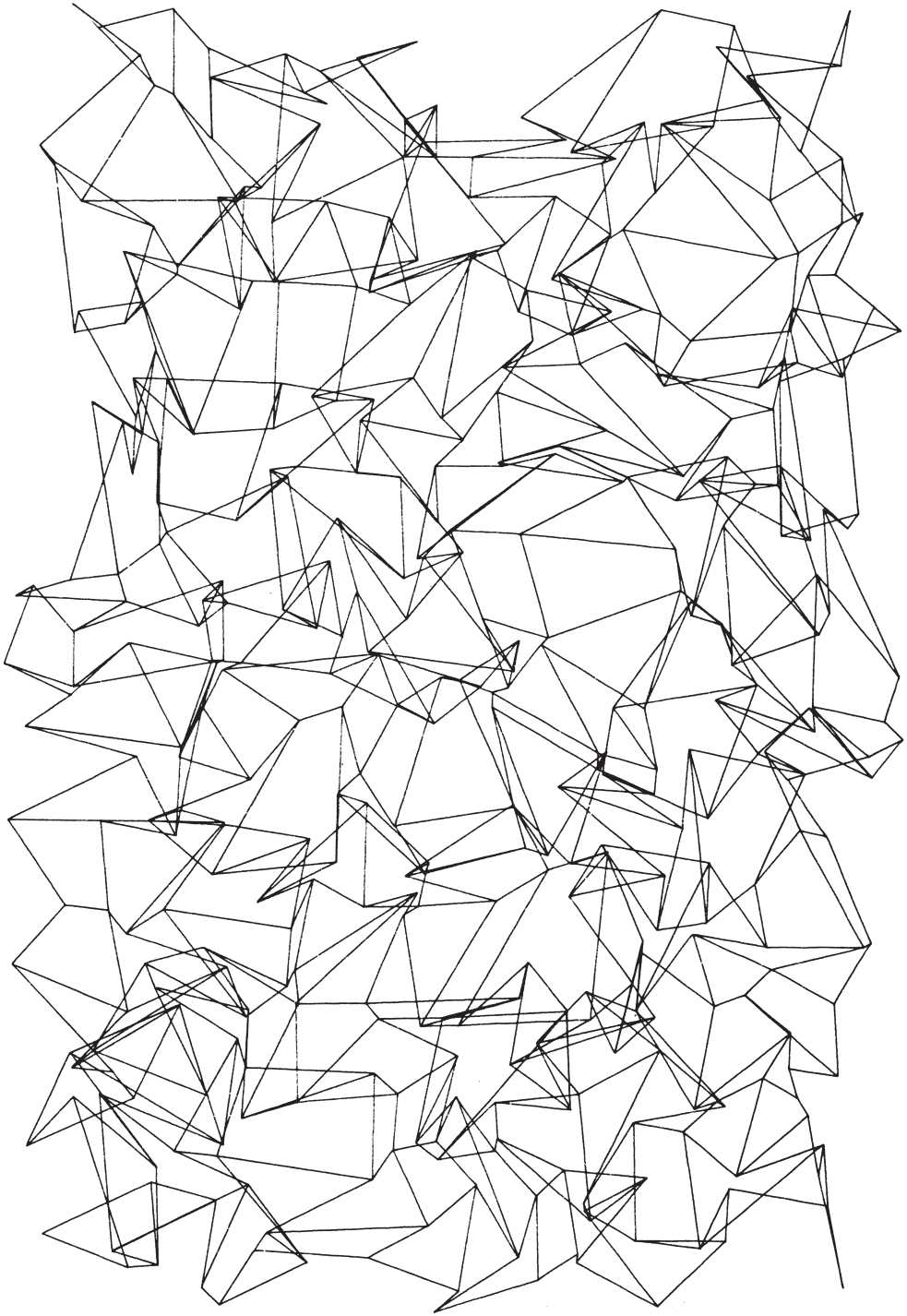


Bild 45

Bild 46



```
18 'END'
19 'END'.,
20 'FOR' P.= −12 'STEP' 1 'UNTIL' 12 'DO'
21 'BEGIN'
22  LEER(A(/P,−8/),B(/P,−8/)).,
23 'FOR' Q.= −7 'STEP' 1 'UNTIL' 8 'DO'
24  LINE(A(/P,Q/),B(/P,Q/))
25 'END'.,
26 'FOR' Q.= −8 'STEP' 1 'UNTIL' 8 'DO'
27 'BEGIN'
28  LEER(A(/12,Q/),B(/12,Q/)).,
29 'FOR' P.=11 'STEP' −1 'UNTIL' −12 'DO'
30  LINE(A(/P,Q/),B(/P,Q/))
31 'END'.,
32  CLOSE
33 'END' GEWEBE STOERUNG ZENTRIERT.,
```

Man analysiert Generator (2) am leichtesten durch zeilenweisen Vergleich mit Prozedur (1). Wir führen dies nicht im einzelnen durch. Es werden, wie in der Prozedur GEWEBE, zunächst zwei Koordinatenmatrizen generiert. Der entsprechende Laufkern einer Schachtelung von zwei Laufanweisungen befindet sich im Fall des Generators (2) in den Zeilen 15 bis 17, wobei die Zeilen 16 und 17 streng der Zeile 9 in (1) entsprechen. Der einzige Unterschied zwischen den Anweisungen in den Zeilen 9 bzw. 16 und 17 besteht in einem *Aufblähungsfaktor* F, mit dem die Zufallsgeneratoren J1 und J2 multipliziert werden. Wie Zeile 15 zeigt, wird die Aufblähung durch die Verkleinerung des Nenners eines arithmetischen Ausdrucks in der Nähe des Koordinatenursprungs hervorgerufen.

Wir schließen diesen Abschnitt mit einer ästhetisch-kosmogonischen Bemerkung. Stellt man sich den Perzi-

pienten am Ende und als Abschluß eines Nachrichtenkanals vor, der ihm ästhetische Information zuspielt, so muß das System aus dem Informationsexpedienten und dem Übertragungskanal komplizierter Zwischenspeicherungen fähig sein. Da zeitlich später generierte Information mit zeitlich früher generierter und gespeicherter im Geneseprozeß *reagiert,* muß die kanalartige Umwelt des Perzipienten als *Informationsreaktionssystem* aufgefaßt werden, in das der Perzipient selbst eingebettet ist.

3.2.10. Ornamente

Die Gewebe, als ästhetische Information aufgefaßt, zeichnen sich durch einen besonders starken Zusammenhang aus, der bereits auf der Expedientenseite gestiftet wird. Kein intelligenter Perzipient ist vorstellbar, der diesen typischen Gewebenexus übersehen oder ihn als perzeptorseitig hinzugefügt auffassen könnte. Im vorliegenden Abschnitt beschäftigen wir uns mit einem Typus ästhetischer Information, der den Geweben insofern nahesteht, als er sich ebenfalls durch hohe Beträge von Makroinnovation auszeichnet. Im Hinblick auf die Stärke des expedientenseitig hergestellten Zusammenhangs jedoch steht der neue Typus am anderen Ende einer Gradation. Wir sprechen von den Flächenornamenten, man betrachte die Bilder 47 und 48.

Besonders interessant ist vielleicht, daß Ornamente durch Zerstörung von Mikroinnovation in Texturen erzeugt werden können. Diese Erzeugungstechnik liefert einen Beweis dafür, daß Ornamente arm an Mikroinnovation sind. Wir wenden uns noch einmal dem Generator (1) aus Abschnitt 3.2.5 zu, der die in den Bildern 29 und 30 gezeigten Texturen erzeugt. Dieser Generator legt mit Hilfe der Prozedur SERIE Irrwege schachbrettartig

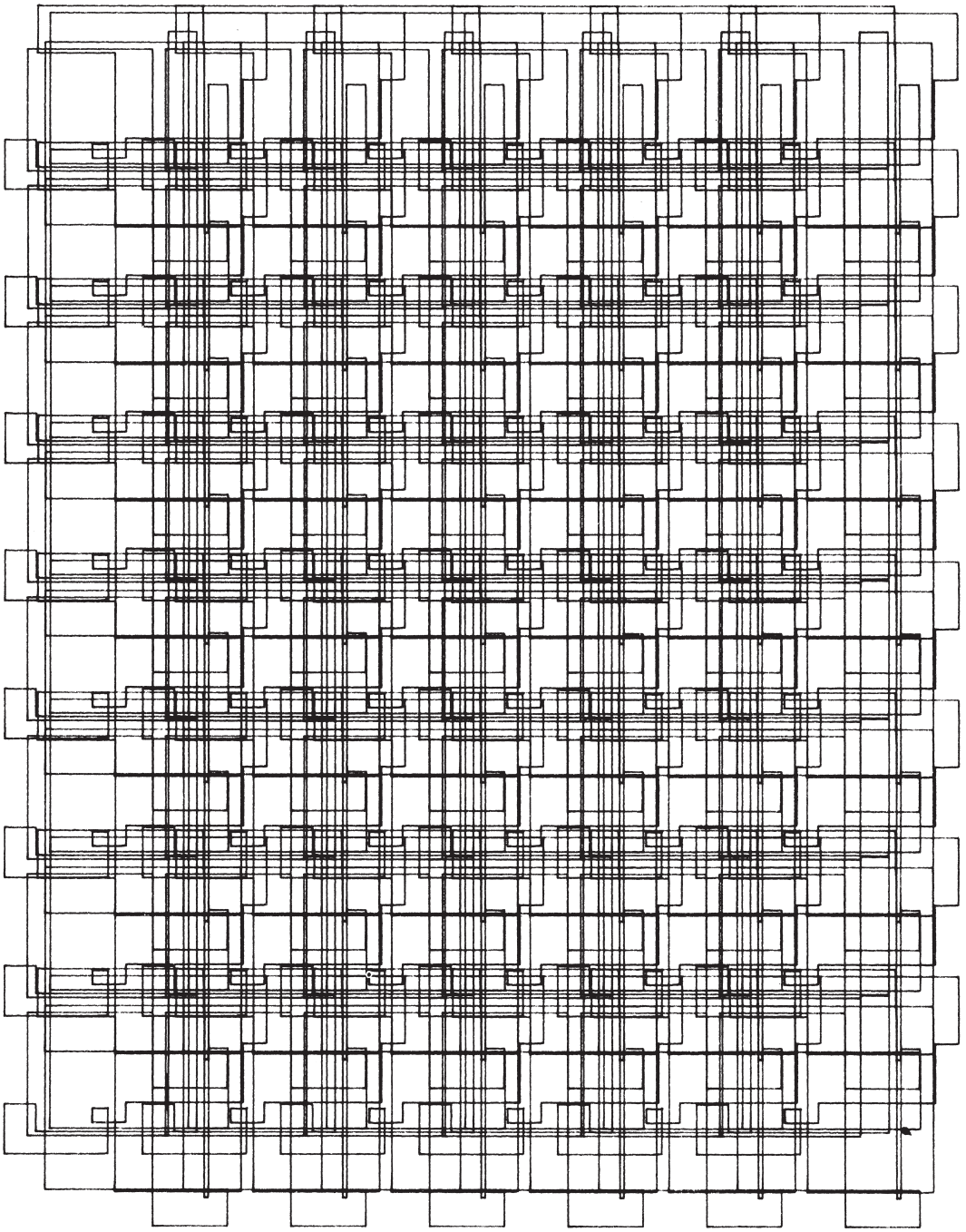
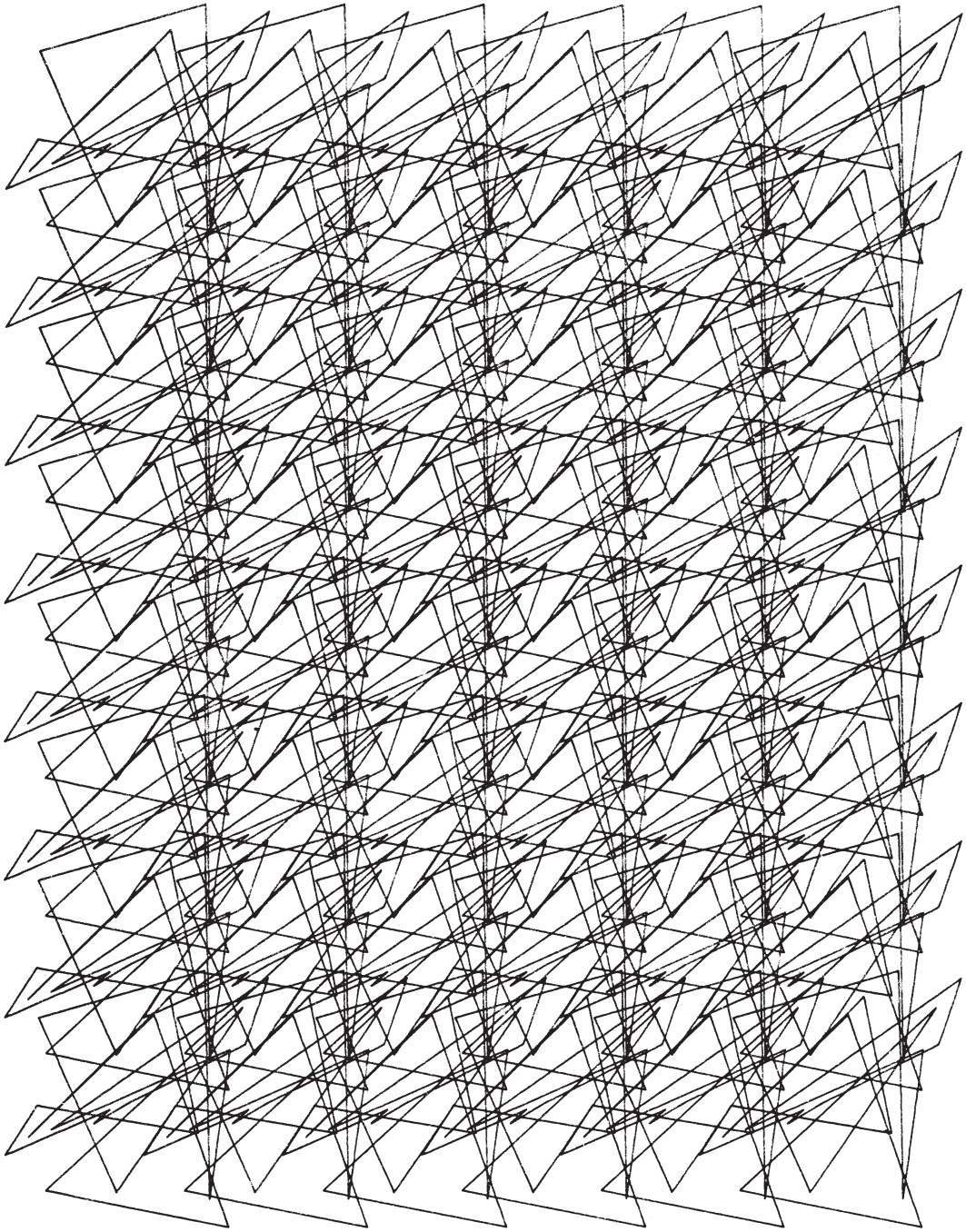


Bild 47

Bild 48



übereinander, die von der Prozedur ELIRR generiert werden. Man kann die von ELIRR herstammenden Irrwege als eine Serie von Eruptionen betrachten. Jede Eruption des Generators ELIRR liefert einen Irrweg, der dem vorhergehenden gleicht, jedoch in keiner Weise mit ihm deckungsgleich ist. Das Fehlen der Deckungsgleichheit rührt davon her, daß die an der Funktion von ELIRR beteiligten Zufallsgeneratoren beim Aufruf nicht neu geladen werden, sondern weiterlaufen und eine Serie neuer Werte liefern. Man betrachte dazu noch einmal den Rumpf von ELIRR, der als Ausdruck (1) in Abschnitt 3.2.2 zu finden ist. In den Zeilen 5, 10 und 12 von (1/3.2.2) stehen die Zufallsgeneratoren J1 und J2. Man kann in die lange Folge der von J1 und J2 errechneten Zufallszahlen dadurch künstlich eine Periode einführen, daß man J1 und J2 bei jedem Aufruf von ELIRR neu initiiert. Am einfachsten erreicht man das dadurch, daß man ELIRR in eine — ebenfalls parameterlose — Rahmenprozedur einfügt, an deren Anfang die Initiation von J1, J2 vorgenommen wird. Nennen wir die Rahmenprozedur ELPER (sozusagen „ELIRR — periodisch"), so kann ELPER folgendermaßen abgefaßt werden:

```
(1)   'PROCEDURE' ELPER.,
      'BEGIN'
       JI1.=JS1., JI2.=JS5., ELIRR
      'END' ELPER.,
```

Aus dem Generator (1/3.2.5) zur Erzeugung der Bilder 29 und 30 läßt sich nun auf sehr einfache Weise ein Generator zur Erzeugung der Bilder 47 und 48 dadurch machen, daß man ELIRR durch ELPER ersetzt.

Die Mikroinnovation der Bilder 47 und 48 ist gleich der Mikroinnovation einer einmaligen Eruption von ELIRR.

Der einzige Irrweg, den ELPER erzeugt, liefert, schachbrettartig übereinandergelegt, die Redundanz — oder den Rapport, wie man allgemein bei Ornamenten sagt — der beiden Flächenornamente. Da sich die kongruenten Irrwege, aus denen die Bilder 47 und 48 aufgebaut sind, gegenseitig überdecken, enthält die Bildredundanz endogene Anteile aus Überlappungseffekten. Jedoch sogar diese endogene Redundanz kehrt, wenigstens im Innern der Graphik, kongruent wieder. Die Originalität der Redundanz des Flächenornaments, die nach Abschnitt 3.1.5 mit der Makroinnovation identisch ist, ist das sichtbare Heraustreten der Schachbrettstruktur. Diese Originalität ist ein Zuwachs an Makroinnovation, wenn man von der Textur herkommt, die durch Anwendung von ELPER periodisiert worden ist. Ornamente sind also, ärmer an Mikroinnovation, reicher an Makroinnovation als Texturen.

Das Grundproblem der generativen Graphik: *Wie bedingt die Struktur des Generators die Struktur seines Produkts?* hat im Fall der Ornamente eine besonders einfache Lösung. Es sind nämlich die Periodizitäten im Bild die sichtbare Spur von periodisch initiierten Zufallsgeneratoren.

Inwiefern stimmt das am Anfang Gesagte: *Im Hinblick auf die Stärke des expedientenseitig hergestellten Zusammenhangs steht das Ornament am Ende einer Gradation?* Nun, es ist anschaulich einzusehen, daß der Nexus des Ornaments mindestens durch eine perzeptorische Funktion nachgeahmt werden kann. Ein drastisches Beispiel ist das willkürliche Übereinanderlegen von *subjektiven Nachbildern* durch mehrfaches Hinblicken auf einen sehr hellen Gegenstand, mit jeweils etwas verändertem Blickwinkel. Ornamente sind ver-

vielfachte Einzelzeichen. Eine derartige Vervielfachung kann der Perzipient mit jedem Zeichen vornehmen, das er perzipiert. Ist aber klar, daß der expedientenseitige Zusammenhang beim Ornament verschwindend klein ist, so ist aus dem vorhergehenden Abschnitt bekannt, daß für Gewebe das Umgekehrte zutrifft. Dazwischen liegt eine Skala von anderen Informationstypen.

3.2.11. Ein Gradationsdiagramm

Der vorliegende Hauptabschnitt 3.2 war dem Versuch gewidmet, eine gewisse Übersicht über das Material der generativen Graphik zu gewinnen. Wir waren uns allerdings von vornherein darüber im klaren, daß eine endgültige Klassifikation graphischer Information nicht möglich ist: Jedes Klassifikationssystem wird gleichzeitig eine Ausgrenzung noch nicht klassifizierter Fälle von Information nach sich ziehen. Diese Erkenntnis ist positiv zu bewerten, da sie nichts anderes als die Kreativität der zur Verfügung stehenden Erzeugungsmöglichkeiten für ästhetische Information bedeutet.

Unsere bewußt als vorläufig angesehene Klassifikation bedient sich der Zerlegung der Klasse generativer Graphiken in Teilklassen, innerhalb derer jeweils eine gewisse, als Gradation bezeichnete, auf- oder absteigende Ordnung möglich ist. Als Ordnungsprinzip in einer Gradation kann der Gehalt an einer Innovationsmodalität oder auch die Dichte des Bildnexus oder noch anderes dienen. Man kann nun die einzelnen Gradationen, die wir schlicht mit den sie tragenden Teilklassen der (offenen) Klasse der generativen Graphiken identifizieren, als Zweige eines Graphen oder Streckenkomplexes auffassen. Diese Darstellungsweise hat den Vorteil, daß die Existenz von disjunkten Gradationen, die an ihren Endpunkten zusammenhängen, anschaulich gemacht wer-

den kann. Das Ergebnis ist ein Netzwerk von Gradationen, oder, wie wir auch sagen können, ein Gradationsdiagramm.

Figur 9 zeigt das Gradationsdiagramm, das unsere Klassifikationsansätze zusammenfaßt. Es enthält auch einige Begriffe, die erst im nächsten Hauptabschnitt über Gestalten und Gegenstände genauer erläutert werden. Wir haben beim Klassifizieren die Gestalten mit Absicht außer acht gelassen, da sie für sich ein Riesenreich von Zeichenkonstellationen aufbauen. Lediglich als Grenzfälle oder Grenzübergänge tauchen die Gestalten im Gradationsdiagramm auf.

Obgleich das Gradationsdiagramm sich eigentlich selbst erläutert, greifen wir einige Eigentümlichkeiten heraus:

1) Der Typus des Haufens bildet einen Endpunkt für mehrere Gradationen, da sowohl Variationenbilder als auch isotrope und anisotrope Texturen durch Zerfall von Anordnung oder Nexus in reine Anhäufungen voneinander unterscheidbarer Einzelzeichen übergehen können.

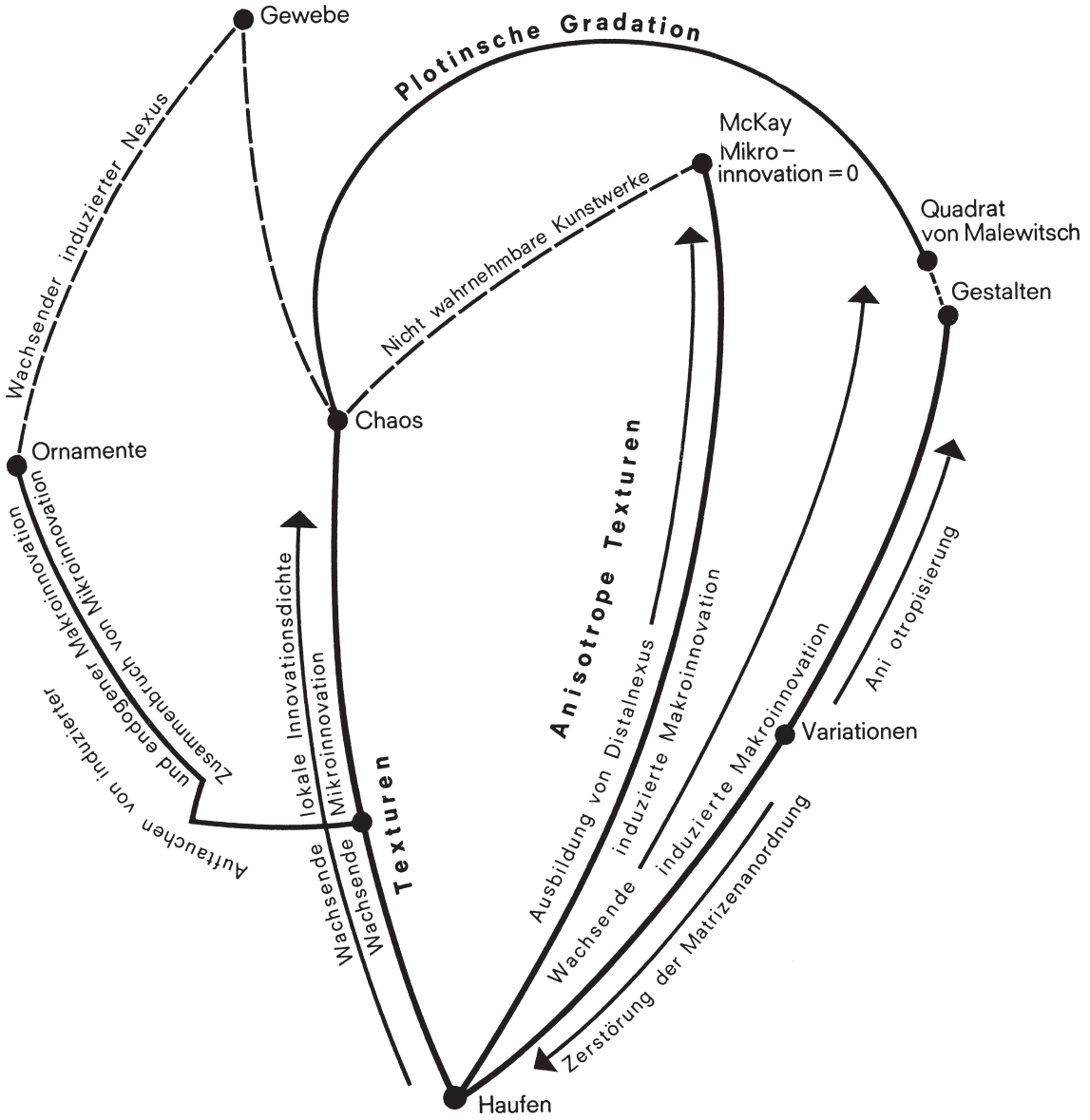
2) Unbegrenztes Anwachsen von Mikroinnovation in isotropen Texturen führt ins Chaos, unbegrenztes Anwachsen von Orientierung in anisotropen Texturen führt zu *idealen Kunstwerken,* beide Extreme stellen Bensesche Grenzfälle ästhetischer Information dar. Fiktiv könnte zwischen den Extremen ein Bereich für den endlichen Intellekt nicht wahrnehmbarer Kunstwerke angenommen werden, so daß drei Punkte mit mittlerer (siehe z. B. Bild 27), unendlich großer und verschwindender Mikroinnovation auf einer geschlossenen Linie lägen.

3) Im Chaos entspringt auch die *Plotinsche Gradation* (siehe nächster Hauptabschnitt), die an diesem Punkt mit unendlichen dichten Texturen der Art der Bilder 49 bis 51 einsetzt. Abgeschlossen denken kann man sich die Plotinsche Gradation mit dem Quadrat von MALEWITSCH, das beispielhaft eine Gestalt ist. Von hier aus schließt sich der Kreis über die Gestalten zu anisotropisierten Variationenbildern (betrachte Bild 26).

4) Die Ornamente sind von den Texturen abzweigbar, allerdings ist ihr Bereich nur durch einen Einbruch der Mikroinnovation erreichbar. Ihr hohes Regelmaß verdanken die Ornamente der räumlich wiederkehrenden Überlagerung kongruenter Elemente. Keine besondere gegenseitige Verknüpfung dieser Elemente ist bei der Bildgenese zur Erzeugung des Regelmaßes notwendig. Der *induzierte Nexus* ist deshalb schwach. Das Gegenteil gilt für die Gewebe, deren induzierter Nexus sogar aus einer Abstimmung später generierter Bildelemente auf früher entstandene hervorgeht.

Die Anlage unseres Gradationsdiagramms ist weitgehend willkürlich, aber diese Willkür entspringt aus dem Wesen der Sache: Der ästhetische Kosmos ist ständig in Evolution, deshalb kann er nicht in ein oder dieses Begriffsnetz eingefangen werden.

Damit beenden wir Hauptabschnitt 3.2. Vermutlich wird eine innige Verbindung und Zusammenarbeit zwischen Informationsästhetik, Wahrnehmungspsychologie und Sinnesphysiologie notwendig werden, um das Klassifikationsproblem erneut und tiefgründiger angreifen zu können. Dabei werden Datenverarbeitungsanlagen mit



Figur 9 Ein Gradationsdiagramm

optisch-rezeptorischer Peripherie als Forschungswerkzeug dienen.

3.3. Gestalt und Gegenstand

Des öfteren haben wir einen naiven Gestaltbegriff benutzt. So sagten wir in Abschnitt 3.1.3 vom zweiten Teilbild von Bild 17, es sei eine echte Gestalt. Schon in Abschnitt 3.1.6 jedoch stießen wir auf einen Informationstyp, der Beziehungen zum Gestaltproblem herstellt und zur Analyse dieses Problems herangezogen werden kann. Wir beschäftigen uns dort mit der inhärenten Makroinnovation und erkannten sie als makroästhetische Auswirkung mikroästhetischer Ursachen. Auch an jener Stelle verwiesen wir auf das zweite Teilbild von Bild 17, dessen endogenes Beziehungs- und Verknüpfungssystem eine Wirkung der Überlagerung von Einzelzeichen ist. Überlagerung tritt auf, deren makroinnovative Konsequenzen unmittelbar von der Funktionalität der Zufallsgeneratoren ausgelöst werden und deshalb in keiner Weise planbar sind.

Die gegenseitige Verknüpfung von Komponentenzeichen durch Überlagerung in einer Konstellation nannten wir gestaltschaffend durch ihre stabilisierende Funktion, wobei Stabilität zunächst keine andere Bedeutung hat als die Festigkeit des Komponentenzusammenhangs. Festigkeit des Komponentenzusammenhangs kommt jedoch auch bei anderen Informationstypen vor, gewiß existiert sie in Texturen und Ornamenten (man beachte noch einmal die Bilder 29 und 47), ein Maximum erreicht sie in den Geweben (siehe Abschnitt 3.2.9). Es muß also noch anderes zur Gestaltgenese beitragen. In Abschnitt 3.1.6 erwähnten wir *großräumige Bildeffekte* als zweite Determinante für Gestalten. Hier nun finden wir eine

Eigenart der Gestalten, die auch von den AE als fundamental angesehen wird (AE, Seite 247): „Das Zeichen als differenziertes Gebilde, als Element kann einem integrierenden ästhetischen Prozeß unterliegen, dann entsteht Ganzheit, Gestalt."

Damit sind wir an einer Stelle angelangt, an der wir die Grenzen der rein generativ vorgehenden Ästhetik bewußt überschreiten und in den Bereich der vollen Kommunikationsästhetik eintreten müssen. Integrierende ästhetische Prozesse können nicht allein durch genetische Abläufe zuwege gebracht werden, sie haben ihren Schwerpunkt nicht auf der Expeditionsseite der Kommunikationskette. Vielmehr ist der Perzeptor ästhetischer Information die integrierende Instanz beim Integrationsprozeß. Man darf nun nicht glauben, daß es beim Ablauf von Integrationsprozessen eine Stelle in der Kommunikationskette gäbe, wo diese aufgebrochen werden könnte, der Integrationsprozeß dabei jedoch ungestört weiterliefe. Ästhetische Kommunikation ist insofern ein fugenloses Ganzes, als der Integrationsprozeß, den sie einschließt, als eine Funktion des auf der Expedientenseite generierten Zeichenstromes existent ist. Allerdings kann sich der Integrationsprozeß ändern, ohne daß der Zeichengeneseprozeß sich ändert, der ja eine wesentliche Voraussetzung des Integrationsprozesses ist. Kunstwerken und anderen Trägern ästhetischer Information ist es eben eigentümlich, daß sie in Kommunikationsprozesse mit funktionell voneinander getrennten Perzipienten eintreten können.

Behalten wir den ganzen Kommunikationsprozeß im Blickfeld, so sind wir allerdings gehalten, zu fragen, ob denn erst die Integration von Zeichenkomplexen zu Gestalten als Perzeptionsaktivität anzunehmen ist, oder ob

solche Aktivität nicht schon auf niedrigerer Organisationshöhe der perzipierten Information wirksam wird. Hier ist es natürlich die inhärente oder endogene Makroinnovation, die zu überdenken ist. In Abschnitt 3.2.6 machten wir ja den Überlappungseffekt für die Endogenese von Makroinnovation verantwortlich. Es ist jedoch nicht zu leugnen, daß die Überlappung und die durch sie bedingte Verhakung von Zeichen in Bildinformationen eine Funktion der Perzeptorstruktur ist. Man sieht das zum Beispiel daran, daß der Überlappungseffekt bei der Analyse von Vexierbildern eine gewisse Plastizität hat und haben muß; diese Plastizität ist sicher auch perzeptorabhängig. Ganz ins Positive gewendet ist die mit dem Überlappungseffekt verbundene Endogenese von Makroinnovation ja eine Form von Genese, von Hervorquellung von Information, wobei die Quellstärke zwar auch vom Expedienten der ästhetischen Information abhängt, jedoch von Perzipient zu Perzipient verschieden sein kann.

Ist also Endogenese von Makroinnovation selbst schon eine Perzipientenaktivität — dabei nach wie vor eine Funktion der expedientenseitig generierten Information, so ist sie jedoch sogar eine Form von Integration.

Was auf dieser Stufe der Integration zustande kommt, ist ein Beziehungs- und Verknüpfungssystem von Bildkomponenten, dessen Basis die Funktionalität der Zufallsgeneratoren ist, dessen Reichhaltigkeit aber von Perzipient zu Perzipient verschieden sein wird. Das am Eingang dieses Abschnitts von Bild 17 Gesagte gilt damit allgemein. Der Komponentenzusammenhang weist Festigkeit auf, die wir weiter oben als Stabilität bezeichneten. Nimmt der Integrationsprozeß, dessen Funktionalität wir jetzt wenigstens in groben Umrissen kennen, eine

bestimmte Modalität an, so entsteht Gestalt. Vage bezeichneten wir in Abschnitt 3.1.6 diese gestalterzeugende Modalität mit dem Ausdruck *„großräumige Bildeffekte"*. Die AE halten jedoch zur Kennzeichnung des gestalterzeugenden Modus von Zeichenintegration ein viel subtileres begriffliches Werkzeug bereit: Die Theorie der Interpretation. Um die betreffenden Stellen in AE verstehen zu können, bedarf es einer Vorbemerkung. Die AE verwenden nämlich neben dem Gestaltbegriff den des Gegenstandes. Wir bringen beide Begriffe dadurch auf einen Nenner, daß wir unter dem Gegenstand die Gestalt selbst, jedoch unter dem Aspekt des gestalterzeugenden Modus von Zeichenintegration, verstehen. Damit wird unzweideutig auf die Verantwortung hingewiesen, die der Perzipient für die Gestaltung des Gegenstandes hat. Der Modus der Gestalterzeugung heißt in den AE *Interpretation* und ist eng verbunden mit Reflexion. Wir benutzen den Begriff *Wahrnehmung* synonym mit *Perzeption* und verstehen unter Wahrnehmungselementen die vom Expedienten angelieferten Zeichen. Dann heißt es in den AE, Seite 249: „Wahrnehmbares ausschließlich durch Wahrnehmbares auszudrücken, bedeutet also in jedem Falle eine Aufgabe der Vorverlagerung: der Zurückführung der Erkenntnis auf die sinnlichen Momente, aus denen sie aufgebaut werden konnte und der Darstellung auf die visuellen Elemente, die sich als Farben und Formen anboten. Damit wird aber offensichtlich der Gegenstand sowohl erkenntnistheoretisch wie ästhetisch sekundär, d. h. keine Frage der unmittelbaren Wahrnehmung, sondern der *Interpretation* der Wahrnehmungselemente."

Der Strategie, „Wahrnehmbares ausschließlich durch Wahrnehmbares auszudrücken", folgten wir z. B. bei der Analyse der Texturen, so, wenn wir in Abschnitt 3.2.6 die

Struktur des Lokalnexus auf wahrnehmbare Protozeichen zurückführten. Auf diese Weise fanden wir beispielsweise in Bild 30 *sternartige Linienzersplitterungen,* die Protozeichen sind. Diese Technik reicht bei der Analyse von Gestalten und Gegenständen nicht mehr aus, vielmehr muß die interpretierende Perzeptionsaktivität in die Untersuchung einbezogen werden, woraus folgt, daß eine vollständige formale Theorie der ästhetischen Kommunikation ein Modell des Perzipienten einzubeziehen hätte. Zur Entstehungsweise des Gegenstandes sagen die AE (Seite 250): „Wir gehen davon aus, daß die Rolle dessen, was wir Gegenstand zu nennen gewohnt sind, in der Kunst, in der Malerei fast paradox erscheint. Sofern nämlich der ästhetische Prozeß, der in der Malerei zum Bild führt, ein reiner echter Wahrnehmungsprozeß ist, arbeitet er zwar mit visuellen Zeichen, wie selbständige Farben und Formen etc., aber er kann in dieser puren Wahrnehmung noch keinen Gegenstand erreichen. Der Gegenstand wird nicht in der bloßen Wahrnehmung gegeben; er entsteht sekundär als ein *deutender, reflektierender Auswahlprozeß,* der an den visuellen Zeichen vorgenommen wird. Indem die Darstellung aus den Wahrnehmungselementen, die als Zeichen fungieren, den Gegenstand erschafft, tritt sie aus der Wahrnehmung in die Interpretation, aus dem Sehen in die Reflexion ein. Durch die Wahrnehmung wird dem ästhetischen Realisationsvorgang also der Gegenstand noch nicht gegeben, erst *durch die Reflexion.*"

Fassen wir zusammen, was wir in dieser Einleitung zum Abschnitt 3.3 über Gestalt und Gegenstand gesagt haben: Auf jeden Fall ereignet sich bei Gegenstandsgenese eine Integration von Zeichenelementen, die mit der Endogenese von Makroinnovation einhergeht. Das Resultat dieses fundamentalen Integrationsprozesses ist ein gestalt-

internes, stabiles Beziehungs- und Verknüpfungssystem von Bildzeichen. Der Integrationsprozeß ist perzeptionsabhängig, sein Ergebnis kann bei manchen Perzipienten reichhaltiger ausfallen als bei den übrigen. Durch einen überlagerten Interpretationsvorgang wird das Integrationsergebnis zusammengefaßt zum Gegenstandsganzen. Daß dieser zweite, sekundäre Integrationsvorgang nicht überflüssig ist, zeigen primäre Integrationsprozesse in Texturen, die lediglich zum Lokalnexus der betreffenden Texturen führen, jedoch keine Gestaltgenese zeitigen. Laut AE ist die interpretierende Integration, die den Gegenstand hervorbringt, verbunden mit einem reflektierenden Auswahlprozeß, der am primären Zeichenensemble stattfindet.

Entsinnen wir uns nun, daß wir uns schon früher, in Abschnitt 3.2.5 an einer Definition der Gestalten versucht haben: *Gestalten sind ästhetische Informationseinheiten mit Lokal- und Distalnexus.* Auf diese Weise unterschieden wir die Gestalten von den Haufen, denen wir Diskontinuität und Distalnexus ohne Lokalnexus zusprachen, und von den Texturen, die sich als Informationskontinua mit Lokalnexus ohne Distalnexus herausstellten. Beide, Lokalnexus und Distalnexus, zeichnen also die Gestalt aus. Sowohl nahe benachbarte, als auch weit entfernte Zeichen haben in ihr miteinander zu tun. Wir müssen jetzt erkennen, daß die Formel *Lokal- und Distalnexus* nichts anderes meint als das stabile, innere Beziehungs- und Verknüpfungssystem der Gestalt, das wir schon mehrfach erwähnten, so daß wir gezwungen sind, jene Formel zu ergänzen: Gestalt ist *Resultat der Gegenstandsreflexion mit Lokal- und Distalnexus.* Was aber ist nun die Reflexion, von der wir hier sprechen, für eine ästhetische Funktion? Wir möchten einen ziemlich spekulativen Lösungsvorschlag für dieses Problem ma-

chen und postulieren, daß man es mit einem System von Reflexionen zu tun hat, die jedoch alle Modulationen einer *gegenstandsschaffenden Urreflexion* sind. Die Urreflexion tritt spontan und selbsttätig ein, wenn eine geeignete Zeichenkonstellation perzipiert wird. Die Erfahrung, daß man allerlei Gestalten in den Wolken erkennen kann, wenn man sie müßig und lange genug betrachtet, geht in unseren Lösungsvorschlag ein.

Im nachfolgenden Abschnitt 3.3.1 bringen wir Bildmaterial zur Gegenstandsreflexion.

3.3.1. Die gegenstandsschaffende Reflexion

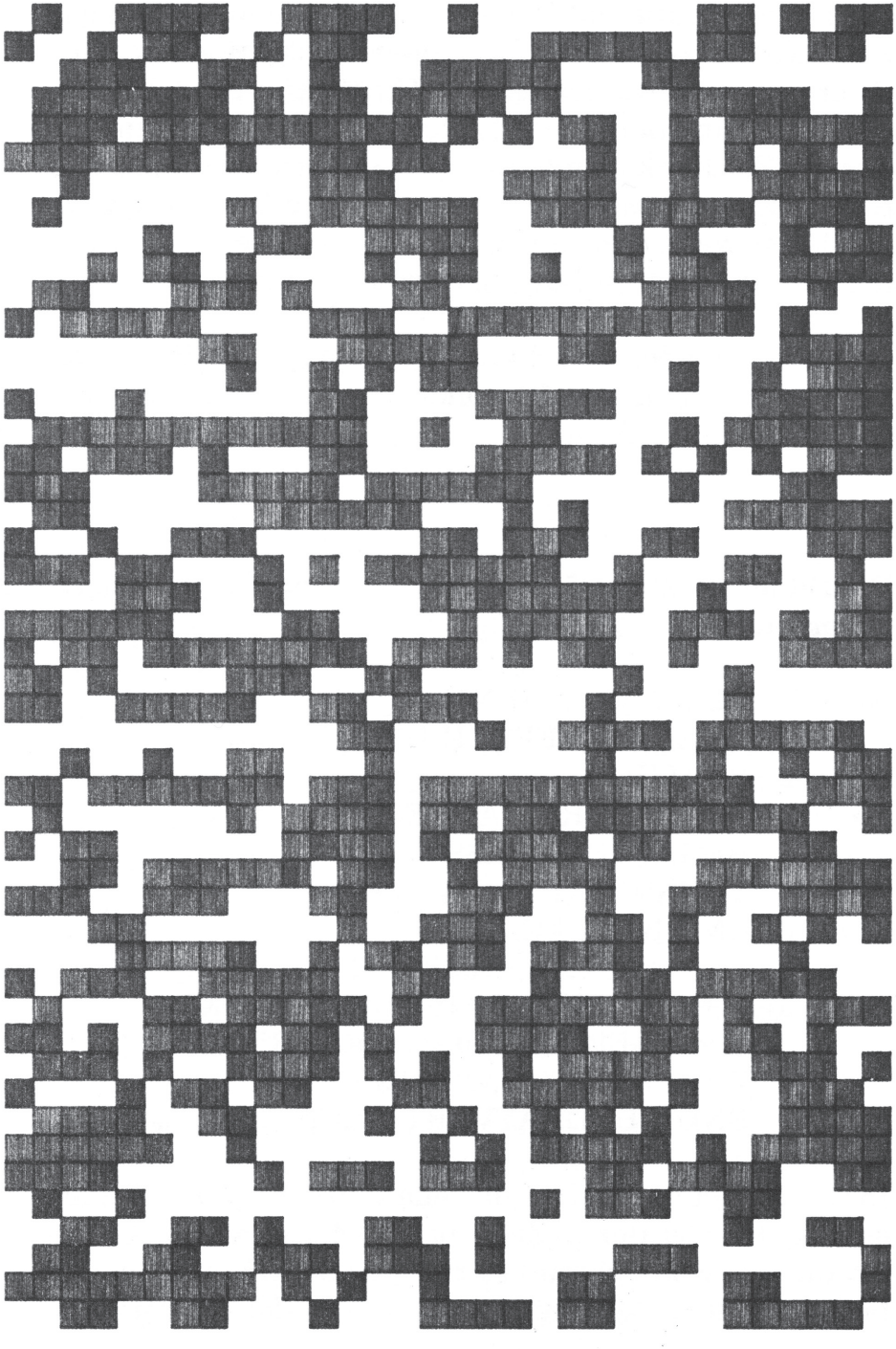
Die Bilder 49, 50 und 51 werden durch folgenden Generator erzeugt:

(1)

```
 1 'BEGIN' COMMENT'
   QUADRATVERTEILUNG.,
 2 'REAL' V.,
 3 'PROCEDURE' RANREC.,
 4 'BEGIN' 'IF' J3 'LESS' .5
 5 'THEN' REC(P,P+V,Q,Q+V,.4)
 6 'END' RANREC.,
 7 JI3.=JS3., JA3.=0., JE3.=1.,
 8 V.=16., SERIE(16.0,16.0,12,8,RANREC).,
 9 V.=8., SERIE(8.0,8.0,24,16,RANREC).,
10 V.=4., SERIE(4.0,4.0,48,32,RANREC)
11 'END' QUADRATVERTEILUNG.,
```

In (1) erzeugt der Aufruf in Zeile 8 Bild 51, der in Zeile 9 erzeugt Bild 50, der in Zeile 10 Bild 49. Verwendung findet die uns wohlbekannte Prozedur SERIE, als Elementarfigur tritt ein Quadrat auf, das entweder schwarz aus-

Bild 49



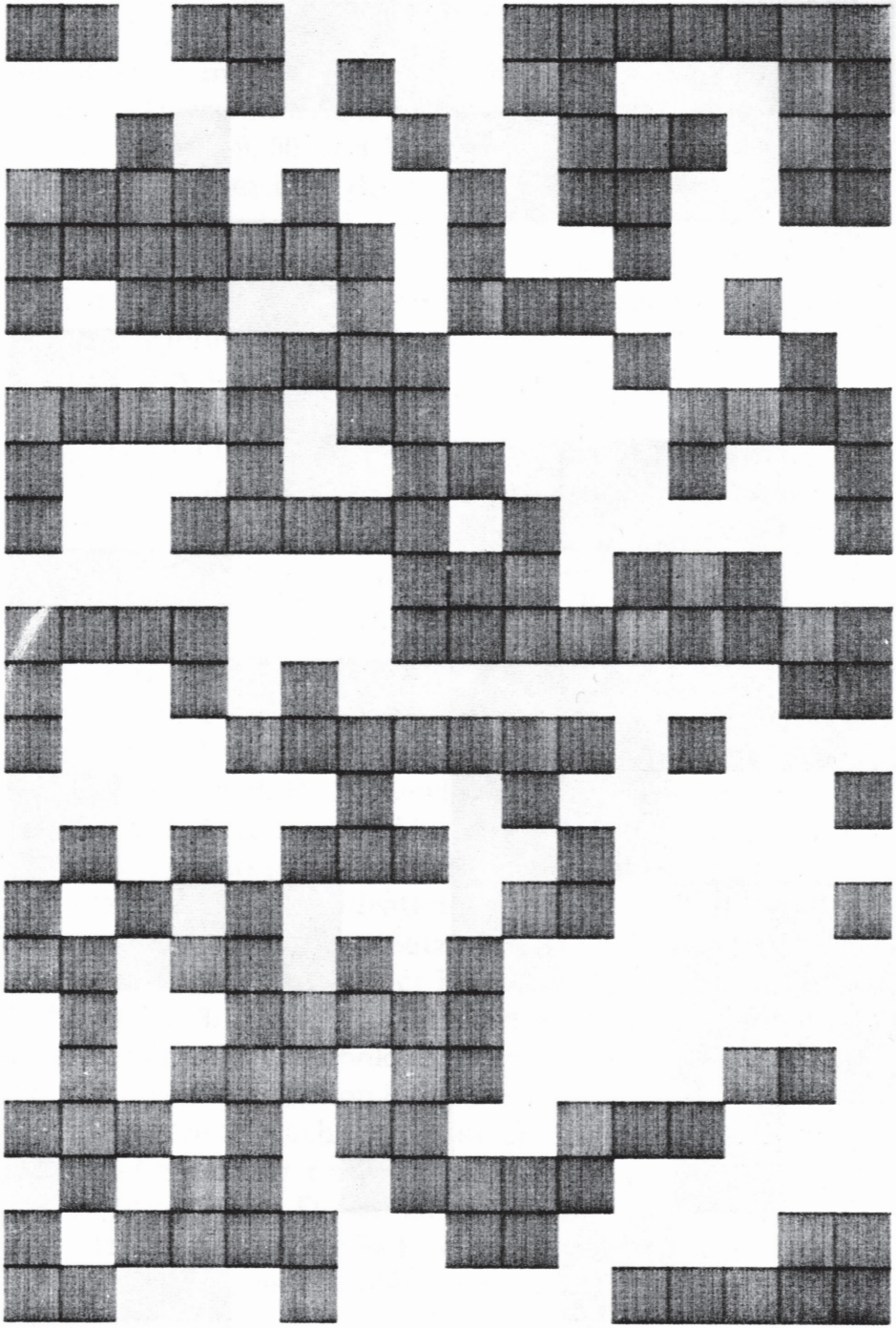
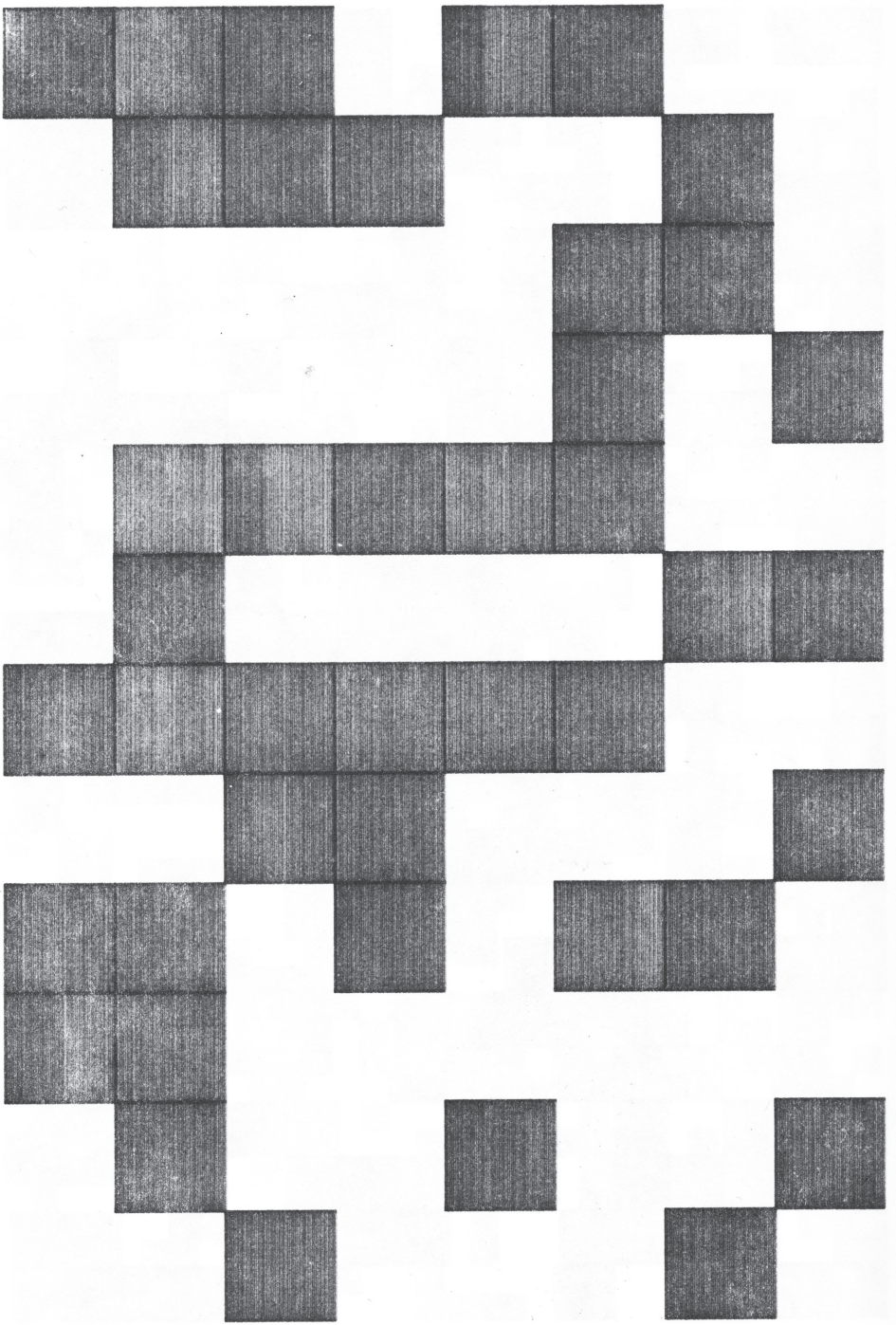


Bild 50

Bild 51



gefüllt oder aber komplementär in weiß erscheint. Die Entscheidung *schwarz oder weiß* fällt der Zufallsgenerator J3 im Rumpf der Prozedur RANREC, die zur Erzeugung der Elementarfigur dient (siehe Zeile 4). Zur Genese der Bilder 49, 50 und 51 würde also auch ein Zufallsgenerator hinreichen, der nur die Werte 0 und 1 liefern kann. Die Prozedur RANREC enthält die Prozedur REC; wir erinnern uns, daß wir die quadratfüllende Prozedur REC ganz am Schluß des ersten Kapitels, in Abschnitt 2.3.2, als Element der Programmiersprache G3 kennengelernt haben.

Man erhält eine sehr streng definierte Gradation von Bildinformationen, wenn man den Parameter V, der in (1) die Breite des Elementarquadrats festlegt, die Folge der Werte 64, 32, 16, 8, 4, 2, 1, 1/2, 1/4, 1/8,... durchlaufen läßt. Das ist so gemeint, daß die Zeilen 8 bis 10 in (1) durch zwei vorausgehende Zeilen für V gleich 64 und 32 und durch eine unendliche Folge von Zeilen für V gleich 2 usw. ergänzt zu denken sind. Der angeführten Wertfolge entsprechend müßten die Aufrufe von SERIE gestaltet werden, so daß z. B. der vorletzte Parameter in SERIE die Folge 2, 4, 8, 16, 32, 64,... zu durchlaufen hätte. Die Gradation beginnt mit einem Raster von sechs schwarzen oder weißen Quadraten und *endet* mit dem einförmig grauen Chaos. Setzt man das Quadrat von Malewitsch an die Spitze der Folge (wir haben dieses Bild in Abschnitt 3.1.5 als Beispiel für die Originalität von Redundanz erwähnt), so bekommt man etwas, was man als die Plotinsche Gradation bezeichnen kann. In der Gradation aufsteigend wächst der Betrag an Mikroinnovation, gleichzeitig wächst natürlich der Betrag an endogener Makroinnovation. Diese Behauptung kann man folgendermaßen belegen: Betrachtet man die Bildinformationen für die Werte V gleich 8 und 4 (die Bilder 50 und

49), um Gestaltetes in ihnen zu entdecken, in gleicher Absicht also, wie man manchmal Wolken betrachtet, so ist die Ausbeute in Bild 49 größer als in Bild 50.

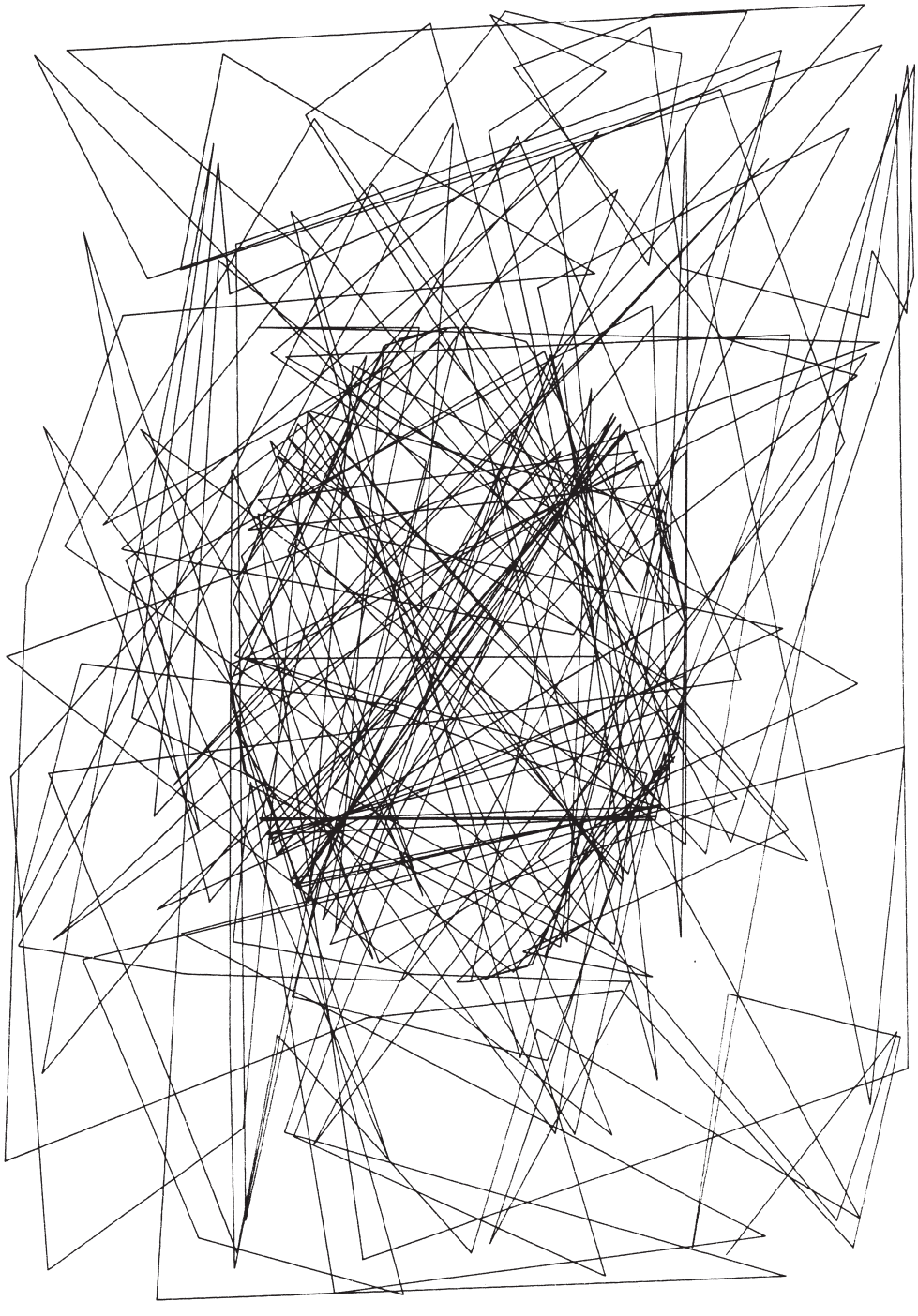
Wie geht das Auffinden von Gestaltetem in den Bildern der Plotinschen Gradation vor sich? Die Antwort ist, daß der Blick des Perzipienten sich einengt und ein begrenztes Feld, ein bestimmtes Quantum an gegliederter Information aus dem Bild herausgreift. Das Selektierte, das Ergebnis des Benseschen *reflektierenden Auswahlprozesses* (siehe das letzte Zitat aus AE), besitzt ein stabilisierendes Beziehungs- und Verknüpfungssystem von Einzelzeichen und Konstellationen. Als anschaulichen Beweis für diese Behauptung benutzen wir Bild 51, das als Resultat des gestaltschaffenden Selektionsprozesses aufgefaßt werden muß. Unsere These ist die, daß die Reflexion, die Bild 51 zu einer Gestalt macht, eine Modulation der Urreflexion ist. Die Urreflexion blickt vom Blickfeld zurück auf ein Etwas, das nicht genauer zu spezifizieren ist, man könnte sagen: auf die *Formalgestalt schlechthin.* An jenes Etwas wird das aus dem Bild Selektierte geheftet, in diesem Augenblick tritt auch die Gestalt ins Dasein. Vom Blickfeld und vom Perzipienten hängt es ab, wie stark die Urreflexion moduliert, d. h. wie reichhaltig das Selektierte gegliedert ist. Daß auch die expedierte Information ein gestaltbestimmender Parameter ist, kann man dadurch zeigen, daß man Bild 51 von verschiedenen Seiten betrachtet. Ist der erste Quadrant (der durch die kleine Eins markiert wird) links oben, so ist die Gestalt am stabilsten. Ist der Quadrant rechts oben, so sieht man eine Gestaltengruppe, so daß der Blick sich weiter einengen muß, um eine stabile Einzelgestalt selektieren zu können.

3.3.2. Kontur und Gestalt

Ein vorgegebenes Feld von Einzelheiten reizt zum Aufsuchen von Gestalten. D. h., daß der gestaltschaffende Selektionsprozeß eine Art von *Aktivität in Wartestellung* besitzt, die ständig darauf aus ist, mit einem Informationsfeld zu reagieren und die Selektion anzubahnen. Im vorhergehenden Abschnitt studierten wir den Fall, in dem ein neutrales Feld von graphischen Zeichen gegeben ist. In einem solchen Feld hebt sich kein Teilbereich von anderen Bereichen derart ab, daß die gestaltschaffende Reflexion eine anfängliche Fixierung erführe und nur zu ergänzen bräuchte, was ihr von der Expedientenseite angeboten wird. Gerade diese Anfangsfixierung ist jedoch im täglichen Leben der häufiger vorliegende Fall, er tritt z. B. ein, wenn uns ein Mensch begegnet, an dem wir entweder achtlos vorübergehen, oder gerade so viel auf ihn achten, daß der Sozialkodex erfüllt wird, den wir jedoch auch mit wachsender Aufmerksamkeit betrachten können, bis hin zur Bannung durch das von uns als wesentlich gestalthaft Erfaßte — die Gestalt ohne jede praktische Bedeutung zunächst.

Bild 52 zeigt einen Modellfall. Jede Initiierung der gestaltschaffenden Reflexion durch die expedierte Information setzt eine derartige Gliederung des Informationsfeldes, der dargebotenen Zeichen-Gesamtkonstellation, voraus. Häufig liegt der Gliederung eine Zerlegung des Feldes in ein Innen- und ein Außengebiet durch eine Randkontur zugrunde. Das Feld gliedert sich in einen neutralen, passiv tragenden Hintergrund, und einen aktiven, den Perzipienten aktivierenden Vordergrund. Wir versuchen uns in diesem Abschnitt nicht an einer Theorie dieser Tatsachen, eine Theorie, die auch wieder den Reaktionsmechanismus des Perzipienten auf ästhetische Information zu modellieren hätte.

Bild 52



Stichwortverzeichnis

aktualisieren 68
Aktualparameter 75
ALGOL 34, 36, 70
Algorithmus 61, 66
Alphabet 157, 170, 174. 240
Anisotropisierung 56, 204, 243, 252
Anweisung 69, 79, 87
Äquivalenzprinzip 68, 71, 110, 174
'ARRAY' 259
Ästhetik, numerische 51
Aufruf einer Prozedur 103
Ausgangsinformation 67, 152
Automat 42, 69, 71, 106, 110
—, interpretierender 91

Bedeutung 6, 10
'BEGIN' 77
Bildgenerator 42, 121, 151
black box 43, 152
Block 69, 116
'BOOLEAN' 115, 138

Chaos 18, 57, 235, 253, 255, 272, 273, 285
CIRC 37
CLOSE 36, 76
Codierung 66
'COMMENT' 97
Computer 7, 11, 31, 48, 66
Computergraphik 8, 10, 11, 145
Computermusik 10
Computerplastik 16

Datenverarbeitungsanlage 25, 66, 273
Deduktion 27
Designserie 18
deterministisch 85
Diskretisation 56
Dispersion 7, 30, 47, 154, 198
Distalnexus 56, 213, 252, 280
Druckeffektor 79
DVA 66
dynamisch 71, 153

Effektor 67, 110
Eingangsinformation 67, 152, 161, 178
Einzelzeichen 7, 160, 271, 286
Elementarfigur 57, 185
ELIRR 55, 189, 221, 269
'END' 77
Entropie 48
'EQUAL' 89
Eröffnung 74, 76, 83

Eruption 171, 200
EXAPT 1, 18
Expedient 11, 277

'FALSE' 140
Flächenornament 59
Formalparameter 75, 106, 185
Funktionsprozedur 115

G 33, 70, 179
G1 70, 80, 110
G2 120
G3 135
Gegenstand 278, 279
Generator 42, 110, 117, 146, 152, 167, 177, 226, 270
Genese 23, 47, 107, 240, 277, 285
Gestalt 59, 164, 181, 206, 210, 213, 273, 275, 279, 287
Gestaltproblem 16, 59
global 58, 116
'GOTO' 90
Gradation 52, 56, 184, 204, 210, 235, 240, 243, 252, 270, 271, 285
Graphik 16, 70, 110, 145
—, generative 23, 24, 48, 184, 213
'GREATER' 89
Gruppierung 56, 201

Haufenbild 56, 206, 213, 243
horizontglobal 116, 120, 151, 260

Information 6, 10, 23, 24, 256
—, ästhetische 6, 8, 11, 16, 24, 145, 253, 266, 272
—, graphische 10, 16, 23, 145
Informationsästhetik 7, 10, 11, 23, 211, 273
Informationskontinuum 56, 213
Informationsreduktion 66
Informationstheorie 65, 151, 182
Informationswandler 66
Innovation 7, 28, 154, 174, 226
Innovationsdichte, lokale 57, 60, 229, 232, 251
'INTEGER' 95, 97, 115
Interpretation 6, 40, 278

Kategorie 43
Kommunikation 6, 8, 27, 276
Kommunikations-ästhetik 61, 276
Komplexität 24, 133, 145
Komposition 11, 44, 153, 157, 160
Kompositor 161, 171

Kontur 56
Koordinatensystem 73
kosmogonisch 58, 265
Kosmos, ästhetischer 40, 58, 273
Kunstsprache 69
Kunstwerk 5, 6, 37, 177, 256, 272, 276
—, perfektes 58
—, reduziertes 168

Laboratorium, ästhetisches 10
Large-Scale-Research 42
Laufanweisung 85, 86, 97, 98, 172, 260
Laufkern 86, 97
LEER 34, 36, 79
'LESS' 89
Light-Pen 62
LINE 34, 36, 79
lokal 116
Lokalnexus 56, 213, 214, 229, 230, 252, 279

makroästhetisch 24, 49, 226, 275
Makroinnovation 45, 180, 236, 249, 252, 253, 266, 270
—, endogene 46, 50, 234, 277
—, exogene 46, 251
—, induzierte 46, 47, 48, 180, 181, 200, 226
—, inhärente 46, 180, 200, 275, 277
Marke 84
metrisch 31
Mikroästhetik 46, 177, 212
mikroästhetisch 24, 49, 275
Mikroinnovation 45, 48, 177, 200, 209, 213, 221, 227, 232, 252, 253, 269, 270
Modell 42, 166, 167, 256

Nachricht 6, 181, 213
Namensaufruf 109
Nexus 27, 198, 270, 272
'NOT' 141
'NOTEQUAL' 89
'NOTGREATER' 89
'NOTLESS' 89

OPEN 36, 74, 75
Operation 16, 23
Operator, multiplikativer 86
Ordnung 24, 29, 271
Ordnungstheorie 51
Ornament 59, 266, 270, 273, 275

Parameter 42, 106
Periodizität 59, 120, 270
Perzeptivität 44, 211
Perzeptor 50, 253

perzeptorisch 41, 227, 276
Perzipient 266, 276, 286, 287
'PROCEDURE' 106, 115
Produkt 46, 156, 270
Programm 8, 11, 24, 25, 48, 67, 69, 145, 167
Programmiersprache 18, 24, 69, 110, 145
Programmverzweigung 84
Protozeichen 60, 229, 232, 236, 279
Prozedur 53, 69, 75, 102, 147, 167, 285
Prozeduraufruf 79
Prozeß 41, 66, 97, 177, 276

Rahmenprogramm 102
Rapport 270
'REAL' 83, 115
Realisation 16, 24, 235
REC 138, 285
Rechteckfläche 36, 134
Redundanz 7, 24, 30, 47, 80, 154, 178, 179, 198, 226, 253
—, endogene 41, 234, 270
Redundanzgenerator 88
Reflexion 6, 23, 278, 279, 286
Regel 16, 23

Schleife 47, 98, 101
Selbststeuerung 88
Selektor 153, 161, 171
selektorisch 44, 159
SERIE 53, 185, 221, 245, 266, 281, 285
Simulation 62
SONK 37, 187
Spezifikation 106
Spontaneität 168
Sprache, formale 11
—, natürliche 11
—, problem-orientierte 34
Sprunganweisung 90
Stabilisierung 59, 275, 286
statisch 71
Stil 10
Struktur 7, 10, 147, 156, 251, 270
—, ästhetische 16, 18, 19, 24
Superzeichen 110
Symmetrie 7
Synthese 10, 11

TAKE 37
Text 145, 157
Textur 56, 211, 232, 252, 255, 275, 280
Theorem 16, 23
Topologie 51
Transduktor 42, 53, 151, 152, 167, 171, 182
'TRUE' 140

Typen, syntaktische 81

Überlagerungseffekt 50, 166, 181
Überlappung 166, 214, 220, 236, 277
Überschiebungseffekt 166, 181
Überschuß 63
Universalautomat 67, 71
'UNTIL' 98
unwahrscheinlich 29, 154
Urreflexion 281, 286

'VALUE' 109
Variable 81
Variationenbild 55, 60, 200, 272
Verbundanweisung 88, 105
Vereinbarung 83, 103
Verhalten 70, 110
Verteilung 16, 23, 24, 153
Vorentscheidung, ästhetische 45, 52, 157

Wahrnehmung 6, 184, 278
Wahrscheinlichkeit 35
Weiche 47, 89, 91, 95, 101, 141, 192
Wertaufruf 106
Wertzuweisung 81, 82
Wortsymbol 77

Zeichen 6, 10, 16, 23, 146, 168, 177, 212
Zeichenautomat 48, 71, 101, 168
Zeicheneffektor 33, 72
Zeichenfunktion 27
Zeichenkonstellation 146, 153, 154, 171, 226, 256, 272, 281
Zeichenmaschine, automatische 7, 10
Zeichenverteilung 7, 23, 168, 198
ZIRK 36, 135
Zufallsgenerator 8, 18, 35, 45, 49, 111, 170, 251, 269, 275, 277
Zusammenhang 24, 59
Zyklus 30, 97

Namensverzeichnis

Anaxagoras 235
Anaximander 235
Ashby, W. Ross 151, 182

Bach, Johann Sebastian 10
Behrenz, P. G. 134
Bense, Max 5, 6, 7, 11, 16, 23, 24, 29, 40, 46, 48, 51, 57, 60, 154, 167, 170, 177, 235, 253, 272, 286
Birkhoff, George D. 110

Demokrit 235

Empedokles 235

Frank, Helmar 110, 255

Gehlen, Arnold 11
Glasauer, Horst 37
Gregory, Richard L. 253

Kandinsky, Wassily 7
Klee, Paul 10
Kupper, H. 10

Manet, Edouard 6, 7
McKay, D. M. 58, 253
Mezei, Leslie 18
Michaux, Henry 168
Moles, Abraham A. 11, 110, 255
Mondrian, Piet 7

Nake, Frieder 5, 10, 18
Nees, Georg 16, 18, 19
Noll, Michael A. 18

Reichenbach, Hans 212

Wiener, Norbert 236

Literaturverzeichnis

Bense, Max

1954 —1960 Aesthetica; 4 Bände. Stuttgart, Krefeld, Baden-Baden.

1965 Aesthetica. Baden-Baden.

1965a Projekte generativer Ästhetik. In: rot 19. Stuttgart.

1967 Semiotik — Allgemeine Theorie der Zeichen. Baden-Baden.

1967a Ästhetik und Programmierung. In: Exakte Ästhetik 5. Stuttgart.

Burkhardt, K. und Maser, S.

1965 rot 24 strukturen-berechnungen. Stuttgart.

Frank, Helmar

1964 Kybernetische Analysen subjektiver Sachverhalte. Quickborn bei Hamburg.

Gehlen, Arnold

1965 Zeitbilder. Frankfurt a. M. — Bonn.

Kandinsky, Wassily

1964 (Zuerst 1926). Punkt und Linie zu Fläche. Bern-Bümpliz.

Kupper, Hubert

1966 Computer und Musikwissenschaft. IBM-Nachrichten 180.

Kupper, Hubert und Görges, Heinz

1967 Computer und Musik. In: Exakte Ästhetik 5. Stuttgart. (Enthält umfangreiche Bibliographie.)

Mezei, Leslie
1964 Artistic Design by Computer. Computers and Automation 13.
1967 Computer Art — a Bibliography. Journal of Computer Studies in the Humanities and Verbal Behaviour 1, No. 1.
1968 Computer Art. Arts Canada, August.

Moles, Abraham A.
1958 Théorie de l'Information et Perception esthétique. Paris.
1967 Über die Verwendung von Rechenanlagen in der Kunst. In: Exakte Ästhetik 5. Stuttgart.

Nake, Frieder
1966 Bemerkung zur Programmierung von Computer-Grafiken. In: Programm-Information PI-21. Deutsches Rechenzentrum, Darmstadt.
1967 Computer-Grafik. In: Exakte Ästhetik 5. Stuttgart.
1968 Erzeugung ästhetischer Objekte mit Rechenanlagen. In: Nichtnumerische Datenverarbeitung, herausgeg. von Rul Gunzenhäuser. Wien — New York.

Nees, Georg
1964 Statistische Grafik. Grundlagenstudien aus Kybernetik und Geisteswissenschaft 5, Heft 3/4.
1964a Variationen von Figuren in der statistischen Grafik. Grundlagenstudien aus Kybernetik und Geisteswissenschaft 5, Heft 3/4.
1967 Programme steuern das Werkzeug. Data Report 2, Heft 4.

1968 Computergraphik und visuelle Komplexität.
BIT 2. Zagreb.
1969 Vom Bit zur dritten Dimension (Interview). Data Report 4, Heft 1.
1969a A graphical simulation- and animation-system for NC-usage.
IFIP PROLAMAT IFAC-Konferenz Rom.
Amsterdam — London.

Nees, Georg und Bense, Max
1965 rot 19 computer-grafik. Stuttgart.

Noll, A. Michael
Art and the Computer. Bell Telephone Laboratories.

1965 Computer Generated Three-Dimensional Movies.
Computers and Automation 14/1965.
1966 Human or Machine: A Subjective Comparison of Piet Mondrian's
„Composition with Lines" (1917) and a Computer Generated Picture.
The Psychological Record, XVI/1966.
1966a Computers and the Visual Arts. Design Quarterly 1966-67.